```
RRRRRRRRRRR      MMM          MMM     SSSSSSSSSSS
RRRRRRRRRRRR     MMM          MMM     SSSSSSSSSSSS
RRRRRRRRRRRR     MMM          MMM     SSSSSSSSSSSS
RRR         RRR  MMMMMM    MMMMMM   SSS
RRR         RRR  MMMMMM    MMMMMM   SSS
RRR         RRR  MMMMMM    MMMMMM   SSS
RRR         RRR  MMM  MMM  MMM  MMM  SSS
RRR         RRR  MMM  MMM  MMM  MMM  SSS
RRR         RRR  MMM  MMM  MMM  MMM  SSS
RRRRRRRRRRRR     MMM          MMM     SSSSSSSSS
RRRRRRRRRRRR     MMM          MMM     SSSSSSSSS
RRRRRRRRRRRR     MMM          MMM     SSSSSSSSS
RRR   RRR        MMM          MMM             SSS
RRR   RRR        MMM          MMM             SSS
RRR   RRR        MMM          MMM             SSS
RRR      RRR     MMM          MMM             SSS
RRR      RRR     MMM          MMM             SSS
RRR      RRR     MMM          MMM             SSS
RRR         RRR  MMM          MMM  SSSSSSSSSSS
RRR         RRR  MMM          MMM  SSSSSSSSSSS
RRR         RRR  MMM          MMM  SSSSSSSSSSS
```

```
RRRRRRRR    MM      MM    333333    RRRRRRRR    RRRRRRRR    VV      VV
RRRRRRRR    MM      MM    333333    RRRRRRRR    RRRRRRRR    VV      VV
RR      RR  MMMM  MMMM  33      33  RR      RR  RR      RR  VV      VV
RR      RR  MMMM  MMMM  33      33  RR      RR  RR      RR  VV      VV
RR      RR  MM  MM  MM        33    RR      RR  RR      RR  VV      VV
RR      RR  MM  MM  MM      33      RR      RR  RR      RR  VV      VV
RRRRRRRR    MM      MM    33        RRRRRRRR    RRRRRRRR    VV      VV
RRRRRRRR    MM      MM    33        RRRRRRRR    RRRRRRRR    VV      VV
RR  RR      MM      MM      33      RR  RR      RR  RR      VV      VV
RR  RR      MM      MM      33      RR  RR      RR  RR      VV      VV
RR    RR    MM      MM  33  33      RR    RR    RR    RR      VV  VV          ....
RR    RR    MM      MM  33  33      RR    RR    RR    RR      VV  VV          ....
RR      RR  MM      MM    333333    RR      RR  RR      RR      VV            ....
RR      RR  MM      MM    333333    RR      RR  RR      RR      VV            ....

LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

```
;     1        0001  0 MODULE RM3RRV (LANGUAGE (BLISS32) ,
;     2        0002  0                  IDENT = 'V04-000'
;     3        0003  0                  ) =
;     4        0004  1 BEGIN
;     5        0005  1 !
;     6        0006  1 !**********************************************************************
;     7        0007  1 !*                                                                    *
;     8        0008  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                          *
;     9        0009  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.           *
;    10        0010  1 !*  ALL RIGHTS RESERVED.                                            *
;    11        0011  1 !*                                                                    *
;    12        0012  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  *
;    13        0013  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
;    14        0014  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER  *
;    15        0015  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  *
;    16        0016  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY  *
;    17        0017  1 !*  TRANSFERRED.                                                       *
;    18        0018  1 !*                                                                    *
;    19        0019  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE  *
;    20        0020  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT  *
;    21        0021  1 !*  CORPORATION.                                                       *
;    22        0022  1 !*                                                                    *
;    23        0023  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS  *
;    24        0024  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.            *
;    25        0025  1 !*                                                                    *
;    26        0026  1 !*                                                                    *
;    27        0027  1 !**********************************************************************
;    28        0028  1
;    29        0029  1
;    30        0030  1 !++
;    31        0031  1 !
;    32        0032  1 ! FACILITY:     RMS32 INDEX SEQUENTIAL FILE ORGANIZATION
;    33        0033  1 !
;    34        0034  1 ! ABSTRACT:
;    35        0035  1 !                ROUTINES TO UPDATE RRV'S
;    36        0036  1 !
;    37        0037  1 !
;    38        0038  1 ! ENVIRONMENT:
;    39        0039  1 !
;    40        0040  1 !               VAX/VMS OPERATING SYSTEM
;    41        0041  1 !
;    42        0042  1 !--
;    43        0043  1
;    44        0044  1 !
;    45        0045  1 ! AUTHOR:        Wendy Koenig        CREATION DATE:             25-JUL-78  15:24
;    46        0046  1 !
;    47        0047  1 ! Modified by:
;    48        0048  1 !
;    49        0049  1 !       V03-012 JWT0149         Jim Teague              19-Jan-1984
;    50        0050  1 !               Correct JWT0146.  Actually, in the event that the new
;    51        0051  1 !               record (for a $PUT) is to be inserted before a deleted
;    52        0052  1 !               record, NXTID should be incremented.  Falling through
;    53        0053  1 !               the logic is correct as long as REC_ADDR is positioned
;    54        0054  1 !               to the next record (just after the deleted record).
;    55        0055  1 !               What was incorrect before was the case where the new
;    56        0056  1 !               record caused a 3-bkt split, and the new record ended
;    57        0057  1 !               up in a bucket of its own (middle bkt).  As rrvs were
```

```
;    58          0058  1 ;                created for the new right bucket, the "if .nxtid nequ 1"
;    59          0059  1 ;                test passed BECAUSE THE NEW RIGHT BUCKET WAS A RECLAIMED
;    60          0060  1 ;                BUCKET!  Thus, nxtid got incremented once too much.
;    61          0061  1 ;                The fix is to remove the "if .nxtid nequ 1" test, because
;    62          0062  1 ;                the rest of the test is quite sufficient to insure correct
;    63          0063  1 ;                id assignment.
;    64          0064  1 ;
;    65          0065  1 ;        V03-011 JWT0146         Jim Teague              05-Dec-1983
;    66          0066  1 ;                Fix an RRV misdirection problem for the case of a
;    67          0067  1 ;                record $PUT before a deleted record.  The record id
;    68          0068  1 ;                of a displaced record was incremented once too much,
;    69          0069  1 ;                because when the record being inserted will end up
;    70          0070  1 ;                in the new bucket, an id is skipped for it when
;    71          0071  1 ;                building RRVs to point to the new bucket.  That's all
;    72          0072  1 ;                cool, but when pos_ins eql rec_addr (the position for
;    73          0073  1 ;                insert is the current record), and the current record
;    74          0074  1 ;                is a deleted record, RMS increments the record id (NXTID)
;    75          0075  1 ;                and then falls almost immediately through to the bottom
;    76          0076  1 ;                of the WHILE loop, where it will increment the new-bucket
;    77          0077  1 ;                record id again.
;    78          0078  1 ;
;    79          0079  1 ;        V03-010 MCN0014         Maria del C. Nasr       22-Mar-1983
;    80          0080  1 ;                More changes in the linkages
;    81          0081  1 ;
;    82          0082  1 ;        V03-009 MCN0013         Maria del C. Nasr       28-Feb-1983
;    83          0083  1 ;                Reorganize linkages
;    84          0084  1 ;
;    85          0085  1 ;        V03-008 TMK0005         Todd M. Katz            27-Jan-1983
;    86          0086  1 ;                Add support for RMS Journalling and RU ROLLBACK Recovery of
;    87          0087  1 ;                ISAM files. This involves adding a flag byte (with one bit
;    88          0088  1 ;                defined - TBL$V_RU_DELETE) to each prologue 3 RRV table entry,
;    89          0089  1 ;                setting the bit within RM$UPDATE_RRV for each entry that refers
;    90          0090  1 ;                to a RU_DELETEd primary data record whose RRV is to be updated,
;    91          0091  1 ;                and referencing the bit within RM$UPDATE_RRV2 before deciding
;    92          0092  1 ;                whether to return an RVU error or not. IT RMS is unable to
;    93          0093  1 ;                position to a RRV and the bit is clear, RMS returns a RVU error
;    94          0094  1 ;                as before. However, if RMS is unable to position to a RRV and
;    95          0095  1 ;                the bit is set, then RMS assumes that the Recovery Unit in
;    96          0096  1 ;                which the RRV was deleted has successfully completed, that the
;    97          0097  1 ;                space occupied by the RRV was reclaimed as part of a general
;    98          0098  1 ;                space reclamation of the bucket, and that there is no need to
;    99          0099  1 ;                return an RVU error in this case.
;   100          0100  1 ;
;   101          0101  1 ;        V03-007 TMK0004         Todd M. Katz            26-Jan-1983
;   102          0102  1 ;                Fix two bugs in RM$UPDATE_RRV.
;   103          0103  1 ;
;   104          0104  1 ;                At one point in this routine a reference was made to a bit in
;   105          0105  1 ;                the current record even though RMS may currently be positioned
;   106          0106  1 ;                to the end of the bucket and there is no current record to
;   107          0107  1 ;                reference. The fix is to make sure that the current record
;   108          0108  1 ;                position is not at the end of the bucket before referencing
;   109          0109  1 ;                this bit.
;   110          0110  1 ;
;   111          0111  1 ;                The second bug is seen in prologue 3 files during $UPDATEs
;   112          0112  1 ;                when the record being updated is in its original bucket and is
;   113          0113  1 ;                to move into a new bucket as the result of the split, and the
;   114          0114  1 ;                record which follows this record in the bucket splitting is
```

```
;  115          0115  1 ;                marked deleted. In this case RMS is not creating a RRV for the
;  116          0116  1 ;                record being modified in the old bucket. To fix this, RMS must
;  117          0117  1 ;                make sure that if it currently is at the position of insertion
;  118          0118  1 ;                of the updated record in its bucket scan, that an RRV is
;  119          0119  1 ;                created for this record in the orginal bucket, if the updated
;  120          0120  1 ;                record was in its original bucket to begin with.
;  121          0121  1 ;
;  122          0122  1 ;        V03-006 TMK0003         Todd M. Katz            10-Jan-1983
;  123          0123  1 ;                In RM$UPDATE_RRV2, always release the scratch buffer that was
;  124          0124  1 ;                used to hold the table of RRVs to be updated. The BDB for this
;  125          0125  1 ;                scratch buffer is to be found in IRB$L_NXTBDB. Formerly this
;  126          0126  1 ;                buffer was bot being released if the data bucket split occurred
;  127          0127  1 ;                because of an $UPDATE and there are old SIDRs to delete;
;  128          0128  1 ;                however, a re-writing of $UPDATE has changed this requirement.
;  129          0129  1 ;
;  130          0130  1 ;        V03-005 KBT0233         Keith B. Thompson       23-Aug-1982
;  131          0131  1 ;                Reorganize psects
;  132          0132  1 ;
;  133          0133  1 ;        V03-004 TMK0002         Todd M. Katz            06-Aug-1982
;  134          0134  1 ;                The RMS cluster solution for next record positioning mandates
;  135          0135  1 ;                that when duplicates are allowed, and a record is deleted,
;  136          0136  1 ;                the space occupied by that record can not be completely
;  137          0137  1 ;                recovered either during the actual deletion of the record
;  138          0138  1 ;                (when the record is just marked deleted, and the space occupied
;  139          0139  1 ;                by the data portion recovered if the file's prologue version
;  140          0140  1 ;                is 3), nor during the space recovery that is attempted when
;  141          0141  1 ;                there is insufficient room in the bucket to accomidate a new
;  142          0142  1 ;                record, or the increased size of an existing record. Therefore,
;  143          0143  1 ;                the routine RM$UPDATE_RRV must be modified, so that RRVS are
;  144          0144  1 ;                never created for deleted records in prologue 3 files, and so
;  145          0145  1 ;                that only deleted RRVs with no RRV pointers are created for
;  146          0146  1 ;                those deleted records in prologue 2 files which are in their
;  147          0147  1 ;                original buckets and require an RRV to preserve their ID from
;  148          0148  1 ;                being recycled.
;  149          0149  1 ;
;  150          0150  1 ;        V03-003 TMK0001         Todd M. Katz            02-Jul-1982
;  151          0151  1 ;                Implement RMS cluster solution for next record positioning.
;  152          0152  1 ;                As the NRP cell has been eliminated and the next record
;  153          0153  1 ;                positioning context is now kept in the IRAB, refer to the
;  154          0154  1 ;                IRAB to obtain the RFA of the new/changed primary data record.
;  155          0155  1 ;                Also, as the module RM3NRP is disappearing, move the routines
;  156          0156  1 ;                RM$CODE_VBN and RM$SELECT_VBN to this module and make them
;  157          0157  1 ;                local routines.
;  158          0158  1 ;
;  159          0159  1 ;        V03-002 MCN0012         Maria del C. Nasr       11-Jun-1982
;  160          0160  1 ;                Eliminate overhead at end of data bucket that was to be
;  161          0161  1 ;                used for duplicate continuation bucket processing.
;  162          0162  1 ;
;  163          0163  1 ;        V03-001 SPR39795        L J Anderson            12-Mar-1982
;  164          0164  1 ;                In the case of a bucket split when run out of IDs, do
;  165          0165  1 ;                NOT update an RRV of a deleted record.  The deleted RRV
;  166          0166  1 ;                has the pointer space squished out, updating the RRV
;  167          0167  1 ;                results in a trashed bucket.
;  168          0168  1 ;
;  169          0169  1 ;        V02-018 KBT0007         K B Thompson            15-Feb-1982
;  170          0170  1 ;                Add code to handle reclaimed bucket next-record-IDs and
;  171          0171  1 ;                add subtitles
```

```
;  172     0172  1 !
;  173     0173  1 !       V02-017 MCN0011         Maria del C. Nasr       28-May-1981
;  174     0174  1 !               More changes required for prologue 3 files.
;  175     0175  1 !
;  176     0176  1 !       V02-016 MCN0006         Maria del C. Nasr       16-Mar-1981
;  177     0177  1 !               Increase size of record identifier to a word in NRP, and
;  178     0178  1 !               other local structures.
;  179     0179  1 !
;  180     0180  1 !       V02-015 REFORMAT        C Saether       01-Aug-1980     22:38
;  181     0181  1 !
;  182     0182  1 !
;  183     0183  1 ! REVISION HISTORY:
;  184     0184  1 !
;  185     0185  1 !   Wendy Koenig,       28-SEP-78  9:11
;  186     0186  1 !   X0002 - SET RRV_ERR ON UPDATE ERROR, AND GO ON TO NEXT RRV
;  187     0187  1 !
;  188     0188  1 !   Wendy Koenig,       29-SEP-78  14:46
;  189     0189  1 !   X0003 - ADJUST POS_INS ON ANY SQUISH, NOT JUST IF BIG_SPLIT
;  190     0190  1 !
;  191     0191  1 !   Christian Saether,  12-OCT-78  12:20
;  192     0192  1 !   X0004 - do not release rrv buffer when in update mode
;  193     0193  1 !
;  194     0194  1 !   Wendy Koenig,       12-OCT-78  14:45
;  195     0195  1 !   X0005 - TAKE ALL THE NRP STUFF OUT OF HERE
;  196     0196  1 !
;  197     0197  1 !   Wendy Koenig,       17-OCT-78  15:40
;  198     0198  1 !   X0006 - CHANGE UPDATE_RRV FOR $UPDATE
;  199     0199  1 !
;  200     0200  1 !   Wendy Koenig,       24-OCT-78  14:03
;  201     0201  1 !   X0007 - MAKE CHANGES CAUSED BY SHARING CONVENTIONS
;  202     0202  1 !
;  203     0203  1 !   Christian Saether,  24-OCT-78  17:38
;  204     0204  1 !   X0008 - give UPDATE_RRV 1 more byte at end of buffer
;  205     0205  1 !
;  206     0206  1 !   Wendy Koenig,       26-OCT-78  11:29
;  207     0207  1 !   X0009 - GET RID OF DEFINITION OF IRC$B_RRV_ID WHICH IS NOW IN THE LIBRARY
;  208     0208  1 !
;  209     0209  1 !   Wendy Koenig,       31-OCT-78  14:09
;  210     0210  1 !   X0010 - FIX BIG, ONLY USE VBN_MID IF BIG_SPLIT
;  211     0211  1 !
;  212     0212  1 !   Christian Saether,  3-NOV-78  8:21
;  213     0213  1 !   X0011 - fix incorrect use of BDB$W_SIZE to BDB$W_NUMB
;  214     0214  1 !
;  215     0215  1 !   Wendy Koenig,       28-NOV-78  11:38
;  216     0216  1 !   X0012 - LOCK BUCKET WHEN UPDATING RRV'S
;  217     0217  1 !
;  218     0218  1 !   Christian Saether,  15-JAN-79  21:41
;  219     0219  1 !   X0013 - eliminate potential deadlock going for rrv's
;  220     0220  1 !
;  221     0221  1 !   Wendy Koenig,       26-JAN-79  9:20
;  222     0222  1 !   X0014 - GET RID OF SETTING VALID
;  223     0223  1 !
;  224     0224  1 !*****
;  225     0225  1
;  226     0226  1 LIBRARY 'RMSLIB:RMS';
;  227     0227  1
;  228     0228  1 REQUIRE 'RMSSRC:RMSIDXDEF';
```

```
;  229          0293  1
;  230          0294  1 ! Define default PSECTS for code.
;  231          0295  1 !
;  232          0296  1 PSECT
;  233          0297  1     CODE = RM$RMS3(PSECT_ATTR),
;  234          0298  1     PLIT = RM$RMS3(PSECT_ATTR);
;  235          0299  1
;  236          0300  1 ! Define some local MACROS.
;  237          0301  1 !
;  238          0302  1 MACRO
;  239          0303  1     IRC$L_RRV_VBN = 3,0,32,0 %,                 ! location of RRV VBN in record
;  240          0304  1     IR3$L_RRV_VBN = 5,0,32,0 %,                 ! new location in prologue 3 files
;  241          0305  1
;  242          0306  1     ! The following macros which define the entries in the local table used for
;  243          0307  1     ! RRV updating, have been reordered to optimize prologue 3 file processing.
;  244          0308  1     ! Those fields that have not changed in size, have been placed up front, so
;  245          0309  1     ! that there are the least possible position variants.  The size of each
;  246          0310  1     ! RRV entry in the table is 10 bytes long for prologue 3 files, and 7 bytes
;  247          0311  1     ! for previous prologue versions.
;  248          0312  1     !
;  249          0313  1     TBL$W_FFB           = 0,0,16,0 %,                   ! stores table size
;  250          0314  1     TBL$B_NEW_VBN       = 0,0,8,0  %,                   ! new VBN index
;  251          0315  1     TBL$L_OLD_VBN       = 1,0,32,0 %,                   ! old VBN value
;  252          0316  1     TBL$B_NEW_ID        = 5,0,8,0  %,                   ! new record id
;  253          0317  1     TBL$W_NEW_ID        = 5,0,16,0 %,                   ! new record id (plg 3)
;  254          0318  1     TBL$B_OLD_ID        = 6,0,8,0  %,                   ! old record id
;  255          0319  1     TBL$W_OLD_ID        = 7,0,16,0 %,                   ! old record id (plg 3)
;  256          0320  1     TBL$B_FLAG          = 9,0,8,0  %,                   ! flag byte (prologue 3)
;  257          0321  1     TBL$V_RU_DELETE     = 9,0,1,0  %,                   ! record is RU_DELETEd
;  258          0322  1
;  259          0323  1     FLG$V_POS_INS = 0,0,1,0 %,
;  260          0324  1     FLG$V_SPLIT_1 = 0,1,1,0 %,
;  261          0325  1     FLG$V_SPLIT_2 = 0,2,1,0 %,
;  262          0326  1     FLG$V_UPD_POS = 0,3,1,0 %,
;  263          0327  1     FLG$V_REC_DEL = 0,4,1,0 %;
;  264          0328  1
;  265          0329  1 ! Linkages.
;  266          0330  1 !
;  267          0331  1 LINKAGE
;  268          0332  1     L_PRESERVE1,
;  269          0333  1     L_RABREG_4567,
;  270          0334  1     L_RABREG_457,
;  271          0335  1     L_RABREG_567,
;  272          0336  1     L_RABREG_67,
;  273          0337  1     L_RELEASE,
;  274          0338  1
;  275          0339  1     ! Local linkages
;  276          0340  1     !
;  277          0341  1     RL$LINKAGE  = JSB() :
;  278          0342  1                   GLOBAL (R_IRAB),
;  279          0343  1     RL$SQUISH   = JSB (REGISTER = 3, REGISTER = 4)
;  280          0344  1                   : GLOBAL (R_REC_ADDR);
;  281          0345  1
;  282          0346  1 ! Forward Routines
;  283          0347  1 !
;  284          0348  1 FORWARD ROUTINE
;  285          0349  1     RM$SQUISH           : RL$SQUISH;
```

```
;  286          0350  1
;  287          0351  1 ! External Routines
;  288          0352  1 !
;  289          0353  1
;  290          0354  1 EXTERNAL ROUTINE
;  291          0355  1     RM$FIND_BY_ID       : RL$RABREG_567,
;  292          0356  1     RM$GETBKT           : RL$RABREG_457,
;  293          0357  1     RM$GETNEXT_REC      : RL$RABREG_67,
;  294          0358  1     RM$RECORD_ID        : RL$RABREG_67,
;  295          0359  1     RM$RECORD_VBN       : RL$PRESERVE1,
;  296          0360  1     RM$RELEASE          : RL$RELEASE_ADDRESSING_MODE( GENERAL ),
;  297          0361  1     RM$RLSBKT           : RL$PRESERVE1;
;  298          0362  1
```

```
;  300     0363  1 %SBTTL 'RM$CODE_VBN'
;  301     0364  1 ROUTINE RM$CODE_VBN (VBN) : RL$LINKAGE =
;  302     0365  1
;  303     0366  1 !++
;  304     0367  1 !
;  305     0368  1 ! FUNCTIONAL DESCRIPTION:
;  306     0369  1 !
;  307     0370  1 ! Converts the new VBN into a 1,2,3 to be stored away temporarily
;  308     0371  1 ! NOTE: CODE_VBN and SELECT_VBN are complimentary routines.
;  309     0372  1 !
;  310     0373  1 ! CALLING SEQUENCE:
;  311     0374  1 !       BSBW RM$CODE_VBN()
;  312     0375  1 !
;  313     0376  1 ! INPUT PARAMETERS:
;  314     0377  1 !       the new VBN
;  315     0378  1 !
;  316     0379  1 ! IMPLICIT INPUTS:
;  317     0380  1 !       IRAB -- VBN_RIGHT, VBN_MID, RFA_VBN
;  318     0381  1 !
;  319     0382  1 ! OUTPUT PARAMETERS:
;  320     0383  1 !       NONE
;  321     0384  1 !
;  322     0385  1 ! IMPLICIT OUTPUTS:
;  323     0386  1 !       NONE
;  324     0387  1 !
;  325     0388  1 ! ROUTINE VALUE:
;  326     0389  1 !       1,2,3
;  327     0390  1 !
;  328     0391  1 ! SIDE EFFECTS:
;  329     0392  1 !       NONE
;  330     0393  1 !
;  331     0394  1 !--
;  332     0395  1
;  333     0396  2     BEGIN
;  334     0397  2
;  335     0398  2     EXTERNAL REGISTER
;  336     0399  2         R_IRAB_STR;
;  337     0400  2
;  338     0401  3     RETURN (
;  339     0402  3
;  340     0403  3         SELECTONE .VBN OF
;  341     0404  3             SET
;  342     0405  3             [.IRAB[IRB$L_VBN_RIGHT]] : 1;
;  343     0406  3             [.IRAB[IRB$L_VBN_MID]] : 2;
;  344     0407  3             [.IRAB[IRB$L_RFA_VBN]] : 3;
;  345     0408  2             TES);
;  346     0409  2
;  347     0410  1     END;                                    ! { end of CODE_VBN }


                                .TITLE  RM3RRV
                                .IDENT  \V04-000\

                                .EXTRN  RM$FIND_BY_ID, RM$GETBKT
                                .EXTRN  RM$GETNEXT_REC, RM$RECORD_ID
                                .EXTRN  RM$RECORD_VBN, RM$RELEASE
                                .EXTRN  RM$RLSBKT
```

```
                                                   .PSECT  RM$RMS3,NOWRT,  GBL,  PIC,2

                   50      04   AE  D0 00000 RM$CODE_VBN:
                                                   MOVL    VBN, R0                          ; 0403
           008C    C9           50  D1 00004       CMPL    R0, 140(IRAB)                    ; 0405
                                04  12 00009       BNEQ    1$
                   50           01  D0 0000B       MOVL    #1, R0
                                05     0000E       RSB
           0090    C9           50  D1 0000F 1$:   CMPL    R0, 144(IRAB)                    ; 0406
                                04  12 00014       BNEQ    2$
                   50           02  D0 00016       MOVL    #2, R0
                                05     00019       RSB
             70    A9           50  D1 0001A 2$:   CMPL    R0, 112(IRAB)                    ; 0407
                                04  13 0001E       BEQL    3$
                   50           01  CE 00020       MNEGL   #1, R0
                                05     00023       RSB
                   50           03  D0 00024 3$:   MOVL    #3, R0
                                05     00027       RSB                                      ; 0410

; Routine Size:  40 bytes,    Routine Base:  RM$RMS3 + 0000

;  348          0411  1
```

```
;  350          0412  1 %SBTTL 'RM$SELECT_VBN'
;  351          0413  1 ROUTINE RM$SELECT_VBN (VALUE, VBN) : RL$LINKAGE =
;  352          0414  1
;  353          0415  1 !++
;  354          0416  1 !
;  355          0417  1 ! FUNCTIONAL DESCRIPTION:
;  356          0418  1 !
;  357          0419  1 ! Converts the 0,1,2,3 which was stored in the RRV table into a relevant VBN.
;  358          0420  1 ! NOTE: CODE_VBN and SELECT_VBN are complimentary routines.
;  359          0421  1 !
;  360          0422  1 ! CALLING SEQUENCE:
;  361          0423  1 !       BSBW RM$SELECT_VBN()
;  362          0424  1 !
;  363          0425  1 ! INPUT PARAMETERS:
;  364          0426  1 !       VALUE -- 0,1,2,3 from the table entry
;  365          0427  1 !       VBN -- if value is 0, VBN is the value we want returned
;  366          0428  1 !
;  367          0429  1 ! IMPLICIT INPUTS:
;  368          0430  1 !       IRAB -- VBN_RIGHT, VBN_MID, RFA_VBN
;  369          0431  1 !
;  370          0432  1 ! OUTPUT PARAMETERS:
;  371          0433  1 !       NONE
;  372          0434  1 !
;  373          0435  1 ! IMPLICIT OUTPUTS:
;  374          0436  1 !       NONE
;  375          0437  1 !
;  376          0438  1 ! ROUTINE VALUE:
;  377          0439  1 !       the actual VBN associated w/ this entry
;  378          0440  1 !
;  379          0441  1 ! SIDE EFFECTS:
;  380          0442  1 !       NONE
;  381          0443  1 !
;  382          0444  1 !--
;  383          0445  1
;  384          0446  2     BEGIN
;  385          0447  2
;  386          0448  2     EXTERNAL REGISTER
;  387          0449  2         R_IRAB_STR;
;  388          0450  2
;  389          0451  3     RETURN (
;  390          0452  3
;  391          0453  3         CASE .VALUE FROM 0 TO 3 OF
;  392          0454  3             SET
;  393          0455  3             [0] : .VBN;
;  394          0456  3             [1] : .IRAB[IRB$L_VBN_RIGHT];
;  395          0457  3             [2] : .IRAB[IRB$L_VBN_MID];
;  396          0458  3             [3] : .IRAB[IRB$L_RFA_VBN];
;  397          0459  2             TES);
;  398          0460  2
;  399          0461  1     END;


                 03            00       04   AE  CF 00000 RM$SELECT_VBN:
                                                         CASEL   VALUE, #0, #3                    ; 0453
```

```
        0019          0013          000D          0008     00005 1$:      .WORD   2$-1$,-                                                  ;
                                                                              3$-1$,-                                                  ;
                                                                              4$-1$,-                                                  ;
                                                                              5$-1$                                                    ;
                                    50      08    AE   D0 0000D 2$:      MOVL    VBN, R0                                          ; 0455
                                                       05 00011           RSB                                                      ;
                                    50    008C    C9   D0 00012 3$:      MOVL    140(IRAB), R0                                    ; 0456
                                                       05 00017           RSB                                                      ;
                                    50    0090    C9   D0 00018 4$:      MOVL    144(IRAB), R0                                    ; 0457
                                                       05 0001D           RSB                                                      ;
                                    50      70    A9   D0 0001E 5$:      MOVL    112(IRAB), R0                                    ; 0458
                                                       05 00022           RSB                                                      ; 0461

; Routine Size:  35 bytes,    Routine Base:  RM$RMS3 + 0028

;  400          0462  1
```

```
;  402          0463  1 %SBTTL  'RM$SQISH'
;  403          0464  1 ROUTINE RM$SQUISH (EOB, SQUISH) : RL$SQUISH =
;  404          0465  1
;  405          0466  1 !++
;  406          0467  1 !
;  407          0468  1 ! FUNCTIONAL DESCRIPTION:
;  408          0469  1 !
;  409          0470  1 ! do the squishing w/o destroying all the registers
;  410          0471  1 !
;  411          0472  1 ! CALLING SEQUENCE:
;  412          0473  1 !       bsbw rm$squish (.eob, .squish);
;  413          0474  1 !
;  414          0475  1 ! INPUT PARAMETERS:
;  415          0476  1 !       eob -- address of end of data to be moved
;  416          0477  1 !       squish -- address of where data is to be moved into
;  417          0478  1 !
;  418          0479  1 ! IMPLICIT INPUTS:
;  419          0480  1 !       rec_addr -- address of beginning of data to be moved
;  420          0481  1 !
;  421          0482  1 ! OUTPUT PARAMETERS:
;  422          0483  1 !       NONE
;  423          0484  1 !
;  424          0485  1 ! IMPLICIT OUTPUTS:
;  425          0486  1 !       NONE
;  426          0487  1 !
;  427          0488  1 ! ROUTINE VALUE:
;  428          0489  1 !       rmssuc always
;  429          0490  1 !
;  430          0491  1 ! SIDE EFFECTS:
;  431          0492  1 !       some data records have been squished out
;  432          0493  1 !
;  433          0494  1 !--
;  434          0495  1
;  435          0496  2     BEGIN
;  436          0497  2
;  437          0498  2     EXTERNAL REGISTER
;  438          0499  2         R_REC_ADDR_STR;
;  439          0500  2
;  440          0501  2     CH$MOVE(.EOB - .REC_ADDR, .REC_ADDR, .SQUISH);
;  441          0502  2     RETURN RMSSUC();
;  442          0503  2
;  443          0504  1     END;                                    ! { end of routine }


                                3C  BB 00000 RM$SQUISH:
                                             PUSHR   #^M<R2,R3,R4,R5>                ; 0464
                        53      56  C2 00002         SUBL2   REC_ADDR, R3            ; 0501
                64      66      53  28 00005         MOVC3   R3, (REC_ADDR), (SQUISH)
                        50      01  D0 00009         MOVL    #1, R0                  ; 0502
                                3C  BA 0000C         POPR    #^M<R2,R3,R4,R5>        ; 0504
                                    05 0000E         RSB                             ;

; Routine Size:  15 bytes,    Routine Base:  RM$RMS3 + 004B
```

```
;  444          0505  1
```

```
;  446          0506  1 %SBTTL  'RM$UPDATE_RRV'
;  447          0507  1 GLOBAL ROUTINE RM$UPDATE_RRV : RL$RABREG_67 NOVALUE =
;  448          0508  1
;  449          0509  1 !++
;  450          0510  1 !
;  451          0511  1 ! FUNCTIONAL DESCRIPTION:
;  452          0512  1 !
;  453          0513  1 !    Create RRV's for records that moved out of this bucket w/o RRV's
;  454          0514  1 !    and make a table so that records that moved before can be updated later.
;  455          0515  1 !    Do not make an entry in the table if the record has been deleted.
;  456          0516  1 !
;  457          0517  1 !    If a deleted record in its orginal bucket is encountered, make a RRV
;  458          0518  1 !    for it if and only if the file's prologue version is not 3, and that RRV
;  459          0519  1 !    is a deleted RRV without a pointer (to reserve the ID so it can not be
;  460          0520  1 !    recycled).
;  461          0521  1 !
;  462          0522  1 ! CALLING SEQUENCE:
;  463          0523  1 !    bsbw rm$update_rrv
;  464          0524  1 !
;  465          0525  1 ! INPUT PARAMETERS:
;  466          0526  1 !    NONE
;  467          0527  1 !
;  468          0528  1 ! IMPLICIT INPUTS:
;  469          0529  1 !    IRAB -- curbdb in irab describing the original bucket
;  470          0530  1 !            nxtbdb describing the extra buffer being used to build the table
;  471          0531  1 !    IDX_DFN - IDX$V_DUPKEYS
;  472          0532  1 !    IFAB - IFB$B_PLG_VER
;  473          0533  1 !
;  474          0534  1 ! OUTPUT PARAMETERS:
;  475          0535  1 !    NONE
;  476          0536  1 !
;  477          0537  1 ! IMPLICIT OUTPUTS:
;  478          0538  1 !    NONE
;  479          0539  1 !
;  480          0540  1 ! ROUTINE VALUE:
;  481          0541  1 !       nothing
;  482          0542  1 !
;  483          0543  1 ! SIDE EFFECTS:
;  484          0544  1 !    The records that were moved out are physically deleted and rrv's are
;  485          0545  1 !       built for all of them.
;  486          0546  1 !    The bucket is marked dirty and valid.
;  487          0547  1 !    Another buffer pointed to by nxtbdb is used to make a table to be used
;  488          0548  1 !       to update rrv's in other buckets.
;  489          0549  1 !    The split points except split itself and possibly pos_ins are destroyed.
;  490          0550  1 !    Those two can still apply to the existing bucket
;  491          0551  1 !    REC_ADDR is destroyed, but it was not an input.
;  492          0552  1 !    Some convuluting stuff is done in the $update case, when there was an
;  493          0553  1 !       original record.
;  494          0554  1 !
;  495          0555  1 !--
;  496          0556  1
;  497          0557  2     BEGIN
;  498          0558  2
;  499          0559  2     EXTERNAL REGISTER
;  500          0560  2         COMMON_RAB_STR,
;  501          0561  2         R_REC_ADDR_STR,
;  502          0562  2         R_IDX_DFN_STR;
```

```
;  503          0563  2
;  504          0564  2      LOCAL
;  505          0565  2          TABLE    : REF BBLOCK,
;  506          0566  2          NXTID    : WORD,
;  507          0567  2          REAL_END       : REF BBLOCK,
;  508          0568  2          EOB      : REF BBLOCK,
;  509          0569  2          SQUISH   : REF BBLOCK,
;  510          0570  2          VBN,
;  511          0571  2          POS_INS : REF BBLOCK,
;  512          0572  2          FLAG     : BLOCK [1],
;  513          0573  2          RRV_VBN,
;  514          0574  2          VBNT,
;  515          0575  2          OLD_ID   : WORD;
;  516          0576  2
;  517          0577  2      GLOBAL REGISTER
;  518          0578  2          R_BKT_ADDR_STR;
;  519          0579  2
;  520          0580  2      FLAG = 0;
;  521          0581  2      TABLE = .BBLOCK[.IRAB[IRB$L_NXTBDB], BDB$L_ADDR] + 2;
;  522          0582  2      BKT_ADDR = .BBLOCK[.IRAB[IRB$L_CURBDB], BDB$L_ADDR];
;  523          0583  2      REC_ADDR = .BKT_ADDR + .IRAB[IRB$W_SPLIT];
;  524          0584  2      EOB = .BKT_ADDR[BKT$W_FREESPACE] + .BKT_ADDR;
;  525          0585  2      REAL_END = .BKT_ADDR + .BBLOCK[.IRAB[IRB$L_CURBDB], BDB$W_NUMB];
;  526          0586  2
;  527          0587  2      ! The real end of the bucket for prologue 3 files is different, since
;  528          0588  2      ! there is some extra information at the end.  The checksum byte is
;  529          0589  2      ! correctly accounted for, so add it back.
;  530          0590  2      !
;  531          0591  2      IF .IFAB[IFB$B_PLG_VER] EQLU PLG$C_VER_3
;  532          0592  2      THEN
;  533          0593  2          REAL_END = .REAL_END - BKT$C_DATBKTOVH + 1;
;  534          0594  2
;  535          0595  2      POS_INS = .BKT_ADDR + .IRAB[IRB$W_POS_INS];
;  536          0596  2      SQUISH = .REC_ADDR;
;  537          0597  2
;  538          0598  2      ! Set Flag Position Insert, if intend on inserting the new record ( or
;  539          0599  2      ! updating the record ) in the old left hand side bucket
;  540          0600  2      !
;  541          0601  2      IF .POS_INS LSSU .REC_ADDR
;  542          0602  2      THEN
;  543          0603  2          FLAG[FLG$V_POS_INS] = 1;
;  544          0604  2
;  545          0605  2      IF .POS_INS EQLU .REC_ADDR
;  546          0606  2          AND
;  547          0607  2          .IRAB[IRB$V_REC_W_LO]
;  548          0608  2      THEN
;  549          0609  2          FLAG[FLG$V_POS_INS] = 1;
;  550          0610  2
;  551          0611  2      ! Set up the starting vbn and the next-record-ID
;  552          0612  2      !
;  553          0613  2      IF .IRAB[IRB$V_BIG_SPLIT]
;  554          0614  2      THEN
;  555          0615  3          BEGIN
;  556          0616  3          VBN = .IRAB [ IRB$L_VBN_MID ];
;  557          0617  3          NXTID = .IRAB [ IRB$W_NID_MID ]
;  558          0618  3          END
;  559          0619  2      ELSE
```

```
;  560     0620  3          BEGIN
;  561     0621  3          VBN = .IRAB [ IRB$L_VBN_RIGHT ];
;  562     0622  3          NXTID = .IRAB [ IRB$W_NID_RIGHT ]
;  563     0623  3          END;
;  564     0624
;  565     0625  2      ! Skip through bucket, deciding where the RRV's for each record should be
;  566     0626  2      ! put -- If in the old (left) bucket, put it at the end of that bucket.
;  567     0627  2      ! If there is an RRV in another bucket, already; then it needs updating,
;  568     0628  2      ! build an entry in the table.  Do not build an entry, if the record has
;  569     0629  2      ! been deleted.
;  570     0630  2      !
;  571     0631  2
;  572     0632  2      WHILE .REC_ADDR LEQU .EOB
;  573     0633  2      DO
;  574     0634  2          BEGIN
;  575     0635  2
;  576     0636  2          BUILTIN
;  577     0637  2              AP;
;  578     0638  2
;  579     0639  2          LOCAL
;  580     0640  2              DIFFERENCE  : WORD;
;  581     0641  2
;  582     0642  2          ! if rec_addr equal to the eob or we're at an rrv (virtual eob ),
;  583     0643  2          ! we still need to do the update for a potential updated record at the
;  584     0644  2          ! eob. but don't do it twice
;  585     0645  2          !
;  586     0646  2
;  587     0647  2          IF .REC_ADDR EQLU .EOB
;  588     0648  2              OR
;  589     0649  2              .REC_ADDR[IRC$V_RRV]
;  590     0650  2          THEN
;  591     0651  2
;  592     0652  2              IF .FLAG[FLG$V_POS_INS]
;  593     0653  2                  OR
;  594     0654  2                  NOT .IRAB[IRB$V_UPDATE]
;  595     0655  2              THEN
;  596     0656  2                  EXITLOOP;
;  597     0657  2
;  598     0658  2          ! If the record is deleted, then save this status in the FLAG byte.
;  599     0659  2          !
;  600     0660  2          IF .REC_ADDR NEQU .EOB
;  601     0661  2              AND
;  602     0662  2              .REC_ADDR[IRC$V_DELETED]
;  603     0663  2          THEN
;  604     0664  2              FLAG[FLG$V_REC_DEL] = 1
;  605     0665  2          ELSE
;  606     0666  2              FLAG[FLG$V_REC_DEL] = 0;
;  607     0667  2
;  608     0668  2          DIFFERENCE = .REC_ADDR - .BKT_ADDR;
;  609     0669  2
;  610     0670  2          ! if more than 1 new bucket, check to see if we've passed a split point
;  611     0671  2          ! if so, the vbn and nxtid have to be changed
;  612     0672  2          !
;  613     0673  2
;  614     0674  2          IF .IRAB[IRB$V_BIG_SPLIT]
;  615     0675  2          THEN
;  616     0676  4              BEGIN
```

```
;  617     0677  4
;  618     0678  4              IF .DIFFERENCE EQLU .IRAB[IRB$W_SPLIT_1]
;  619     0679  4                  AND
;  620     0680  4                  NOT .FLAG[FLG$V_SPLIT_1]
;  621     0681  4              THEN
;  622     0682  4
;  623     0683  5                  IF (.FLAG[FLG$V_POS_INS]
;  624     0684  5                      OR
;  625     0685  5                      NOT .IRAB[IRB$V_REC_W_LO])
;  626     0686  4                      OR
;  627     0687  4                      NOT .IRAB[IRB$V_UPDATE]
;  628     0688  4                  THEN
;  629     0689  5                      BEGIN
;  630     0690  5                      FLAG[FLG$V_SPLIT_1] = 1;
;  631     0691  5
;  632     0692  5                      ! Use the RFA bucket
;  633     0693  5                      !
;  634     0694  5                      VBN = .IRAB [ IRB$L_RFA_VBN ];
;  635     0695  5
;  636     0696  5                      ! If there is no RFA bucket then use the right bucket
;  637     0697  5                      ! else its ok to use the RFA bucket and next-record-ID
;  638     0698  5                      !
;  639     0699  5                      IF .VBN EQLU 0
;  640     0700  5                      THEN
;  641     0701  6                          BEGIN
;  642     0702  6                          VBN = .IRAB [ IRB$L_VBN_RIGHT ];
;  643     0703  6                          NXTID = .IRAB [ IRB$W_NID_RIGHT ]
;  644     0704  6                          END
;  645     0705  5                      ELSE
;  646     0706  5                          NXTID = .IRAB [ IRB$W_RFA_NID ]
;  647     0707  5
;  648     0708  4                      END;
;  649     0709  4
;  650     0710  4                  IF .DIFFERENCE EQLU .IRAB[IRB$W_SPLIT_2]
;  651     0711  4                      AND
;  652     0712  4                      NOT .FLAG[FLG$V_SPLIT_2]
;  653     0713  4                  THEN
;  654     0714  5                      BEGIN
;  655     0715  5
;  656     0716  5                      FLAG [ FLG$V_SPLIT_2 ] = 1;
;  657     0717  5
;  658     0718  5                      VBN = .IRAB [ IRB$L_VBN_RIGHT ];
;  659     0719  5                      NXTID = .IRAB [ IRB$W_NID_RIGHT ]
;  660     0720  5
;  661     0721  4                      END;
;  662     0722  4
;  663     0723  3                  END;
;  664     0724  3
;  665     0725  3              ! if this is the pos for insert, and the record really and truly
;  666     0726  3              ! belongs here, increment the nxtid but make sure that we can never
;  667     0727  3              ! come back to pos_ins more than once if this is an update and the
;  668     0728  3              ! record belonged in the middle bkt all by itself, set up vbn1  to
;  669     0729  3              ! indicate such
;  670     0730  3              !
;  671     0731  3              VBN1 = .VBN;
;  672     0732  3
;  673     0733  3              IF .REC_ADDR EQLU .POS_INS
```

```
;  674     0734  3            AND
;  675     0735  3            NOT .FLAG[FLG$V_POS_INS]
;  676     0736  3        THEN
;  677     0737  4            BEGIN
;  678     0738  4            FLAG[FLG$V_POS_INS] = 1;
;  679     0739  4
;  680     0740  4            IF .IRAB[IRB$V_UPDATE]
;  681     0741  4            THEN
;  682     0742  5                BEGIN
;  683     0743  5                FLAG[FLG$V_UPD_POS] = 1;
;  684     0744  5
;  685     0745  5                IF .IRAB[IRB$V_BIG_SPLIT]
;  686     0746  5                    AND
;  687     0747  6                    (.IRAB[IRB$W_SPLIT] EQLU .IRAB[IRB$W_SPLIT_1])
;  688     0748  5                THEN
;  689     0749  6                    BEGIN
;  690     0750  6                    FLAG[FLG$V_SPLIT_1] = 0;
;  691     0751  6                    VBN1 = .IRAB[IRB$L_VBN_MID]
;  692     0752  6                    END
;  693     0753  5                END
;  694     0754  5
;  695     0755  4            ELSE
;  696     0756  5                BEGIN
;  697     0757  5                !
;  698     0758  5                ! Ok, here's the scoop on what's going down here:
;  699     0759  5                !
;  700     0760  5                ! If this is the position for insert, AND the new
;  701     0761  5                ! record doesn't go into a bucket all by itself
;  702     0762  5                ! (i.e., a 3-bkt split), AND the new record doesn't
;  703     0763  5                ! go into the old bucket, then skip an id to account
;  704     0764  5                ! for the id taken up by the new record when it winds
;  705     0765  5                ! up in the new bucket.
;  706     0766  5                !
;  707     0767  5                IF .IRAB[IRB$W_SPLIT] NEQU .IRAB[IRB$W_SPLIT_1]
;  708     0768  5                    AND
;  709     0769  5                NOT .IRAB[IRB$V_REC_W_LO]
;  710     0770  5
;  711     0771  5                THEN
;  712     0772  5                    NXTID = .NXTID + 1
;  713     0773  5                END
;  714     0774  3            END;
;  715     0775  3
;  716     0776  3        AP = 3;
;  717     0777  3
;  718     0778  4        BEGIN
;  719     0779  4
;  720     0780  4        GLOBAL REGISTER
;  721     0781  4            R_BDB;
;  722     0782  4
;  723     0783  4        IF .FLAG[FLG$V_UPD_POS]
;  724     0784  4        THEN
;  725     0785  4            RRV_VBN = .IRAB[IRB$L_PUTUP_VBN]
;  726     0786  4        ELSE
;  727     0787  4            RRV_VBN = RM$RECORD_VBN();
;  728     0788  3        END;
;  729     0789  3
;  730     0790  3        ! if the VBN's are equal, then this record has never moved and, thus
```

```
;  731     0791  3          ! it needs an RRV; otherwise, it has an RRV elsewhere. NOTE that there
;  732     0792  3          ! is no need to create an RRV for this record (even if the the VBNs
;  733     0793  3          ! are equal) if the record is deleted and the file is a prologue 3
;  734     0794  3          ! file.
;  735     0795  3          !
;  736     0796  3          IF .RRV_VBN EQLU .BBLOCK[.IRAB[IRB$L_CURBDB], BDB$L_VBN]
;  737     0797  3             AND
;  738     0798  5             (NOT (.IFAB[IFB$B_PLG_VER] GEQU PLG$C_VER_3
;  739     0799  5                         AND
;  740     0800  5                        .FLAG[FLG$V_REC_DEL])
;  741     0801  4                  OR
;  742     0802  4                 .FLAG[FLG$V_UPD_POS])
;  743     0803  3          THEN
;  744     0804  4              BEGIN
;  745     0805  4
;  746     0806  4              LOCAL
;  747     0807  4                  RRV_SIZE;
;  748     0808  4
;  749     0809  4              IF .FLAG[FLG$V_UPD_POS]
;  750     0810  4              THEN
;  751     0811  4                  OLD_ID = .IRAB[IRB$W_PUTUP_ID]
;  752     0812  4              ELSE
;  753     0813  4                  OLD_ID = RMSRECORD_ID();
;  754     0814  4
;  755     0815  4              IF .IFAB[IFB$B_PLG_VER] LSSU PLG$C_VER_3
;  756     0816  4              THEN
;  757     0817  4                  IF NOT .FLAG[FLG$V_REC_DEL]
;  758     0818  4                  THEN
;  759     0819  4                      RRV_SIZE = 7
;  760     0820  4                  ELSE
;  761     0821  4                      RRV_SIZE = 2
;  762     0822  4              ELSE
;  763     0823  4                  RRV_SIZE = 9;
;  764     0824  4
;  765     0825  4              ! if there is not enough physical room at the end of the bucket to
;  766     0826  4              ! build an rrv, make enough
;  767     0827  4              !
;  768     0828  4              IF (.EOB + .RRV_SIZE) GEQU .REAL_END
;  769     0829  4              THEN
;  770     0830  5                  BEGIN
;  771     0831  5
;  772     0832  5                  IF NOT .FLAG[FLG$V_UPD_POS]
;  773     0833  5                  THEN
;  774     0834  5                      RMSGETNEXT_REC();
;  775     0835  5
;  776     0836  5                  RMSSQUISH(.EOB, .SQUISH);
;  777     0837  5                  EOB = .EOB - (.REC_ADDR - .SQUISH);
;  778     0838  5
;  779     0839  5                  ! unfortunately, if we squish records out, we also have to
;  780     0840  5                  ! update all the pointers to the bucket
;  781     0841  5                  !
;  782     0842  5
;  783     0843  5                  IF .IRAB[IRB$V_BIG_SPLIT]
;  784     0844  5                  THEN
;  785     0845  6                      BEGIN
;  786     0846  6
;  787     0847  6                      IF .SQUISH LEQU .BKT_ADDR + .IRAB[IRB$W_SPLIT_1]
```

```
;  788     0848  6                THEN
;  789     0849  7                    BEGIN
;  790     0850  7
;  791     0851  7                    IF .BKT_ADDR + .IRAB[IRB$W_SPLIT_1] LEQU .REC_ADDR
;  792     0852  7                    THEN
;  793     0853  7                        IRAB[IRB$W_SPLIT_1] = .SQUISH - .BKT_ADDR
;  794     0854  7                    ELSE
;  795     0855  7                        IRAB[IRB$W_SPLIT_1] = .IRAB[IRB$W_SPLIT_1] -
;  796     0856  8                        (.REC_ADDR - .SQUISH)
;  797     0857  6                    END;
;  798     0858  6
;  799     0859  6                IF .SQUISH LEQU .BKT_ADDR + .IRAB[IRB$W_SPLIT_2]
;  800     0860  6                THEN
;  801     0861  7                    BEGIN
;  802     0862  7
;  803     0863  7                    IF .BKT_ADDR + .IRAB[IRB$W_SPLIT_2] LEQU .REC_ADDR
;  804     0864  7                    THEN
;  805     0865  7                        IRAB[IRB$W_SPLIT_2] = .SQUISH - .BKT_ADDR
;  806     0866  7                    ELSE
;  807     0867  7                        IRAB[IRB$W_SPLIT_2] = .IRAB[IRB$W_SPLIT_2] -
;  808     0868  8                        (.REC_ADDR - .SQUISH)
;  809     0869  6                    END;
;  810     0870  6
;  811     0871  5                END;
;  812     0872  5
;  813     0873  5            IF .SQUISH LEQU .POS_INS
;  814     0874  5            THEN
;  815     0875  6                BEGIN
;  816     0876  6
;  817     0877  6                IF .POS_INS LEQU .REC_ADDR
;  818     0878  6                THEN
;  819     0879  6                    POS_INS = .SQUISH
;  820     0880  6                ELSE
;  821     0881  7                    POS_INS = .POS_INS - (.REC_ADDR - .SQUISH)
;  822     0882  5                END;
;  823     0883  5
;  824     0884  5                REC_ADDR = .SQUISH;
;  825     0885  5                END
;  826     0886  5
;  827     0887  5            ! Else we do not have to squish a record out.
;  828     0888  5            !
;  829     0889  4            ELSE
;  830     0890  4                IF NOT .FLAG[FLG$V_UPD_POS]
;  831     0891  4                THEN
;  832     0892  4                    RM$GETNEXT_REC();
;  833     0893  4
;  834     0894  4            ! Build the RRV at the end of the bucket and update EOB
;  835     0895  4            !
;  836     0896  4            EOB[IRC$B_CONTROL] = 0;
;  837     0897  4            EOB[IRC$V_RRV] = 1;
;  838     0898  4
;  839     0899  4            IF .IFAB[IFB$B_PLG_VER] LSSU PLG$C_VER_3
;  840     0900  4            THEN
;  841     0901  4
;  842     0902  4                ! If the record is deleted and the file is not a prologue 3
;  843     0903  4                ! file then created a two-byte deleted RRV for the record.
;  844     0904  4                !
```

```
;   845     0905  4                   IF .FLAG[FLG$V_REC_DEL]
;   846     0906  4                   THEN
;   847     0907  5                       BEGIN
;   848     0908  5                       EOB[IRC$V_NOPTRSZ] = 1;
;   849     0909  5                       EOB[IRC$V_DELETED] = 1;
;   850     0910  5                       EOB[IRC$B_ID] = .OLD_ID;
;   851     0911  5                       EOB = .EOB + 2;
;   852     0912  5                       END
;   853     0913  4                   ELSE
;   854     0914  5                       BEGIN
;   855     0915  5                       EOB[IRC$V_PTRSZ] = 2;
;   856     0916  5                       EOB[IRC$B_ID] = .OLD_ID;
;   857     0917  5                       EOB[IRC$B_RRV_ID] = .NXTID;
;   858     0918  5                       EOB[IRC$L_RRV_VBN] = .VBN1;
;   859     0919  5                       EOB = .EOB  + $BYTEOFFSET(IRC$L_RRV_VBN)
;   860     0920  5                                   + $BYTESIZE(IRC$L_RRV_VBN);
;   861     0921  5                       END
;   862     0922  4               ELSE
;   863     0923  5                   BEGIN
;   864     0924  5                   EOB[IRC$V_PTRSZ] = 2;
;   865     0925  5                   EOB[IRC$W_ID] = .OLD_ID;
;   866     0926  5                   EOB[IRC$W_RRV_ID] = .NXTID;
;   867     0927  5                   EOB[IR3$L_RRV_VBN] = .VBN1;
;   868     0928  5                   EOB = .EOB + $BYTEOFFSET(IR3$L_RRV_VBN)
;   869     0929  5                              + $BYTESIZE(IR3$L_RRV_VBN);
;   870     0930  4                   END;
;   871     0931  4               END
;   872     0932  4
;   873     0933  4           ! the record has moved before, so make an entry in the table so we can
;   874     0934  4           ! update the record's old RRV, later.  Make an entry only if the record
;   875     0935  4           ! is present (ie, do not update deleted RRV's).  The only time there will
;   876     0936  4           ! be a deleted record in the middle of the bucket, is if this split is
;   877     0937  4           ! happening because of no more id's available (not because of lack of
;   878     0938  4           ! space).  In this case, the routine to squish the deleted records out
;   879     0939  4           ! of the bucket is not called, as space is not the problem.
;   880     0940  4           !
;   881     0941  4           !
;   882     0942  3           ELSE
;   883     0943  3               IF NOT .FLAG[FLG$V_REC_DEL]
;   884     0944  3               THEN
;   885     0945  4                   BEGIN
;   886     0946  4                   TABLE[TBL$B_NEW_VBN] = RM$CODE_VBN(.VBN1);
;   887     0947  4                   TABLE[TBL$L_OLD_VBN] = .RRV_VBN;
;   888     0948  4
;   889     0949  4                   IF .IFAB[IFB$B_PLG_VER] LSSU PLG$C_VER_3
;   890     0950  4                   THEN
;   891     0951  5                       BEGIN
;   892     0952  5                       TABLE[TBL$B_NEW_ID] = .NXTID;
;   893     0953  5
;   894     0954  5                       IF .FLAG[FLG$V_UPD_POS]
;   895     0955  5                       THEN
;   896     0956  5                           TABLE[TBL$B_OLD_ID] = .IRAB[IRB$W_PUTUP_ID]
;   897     0957  5                       ELSE
;   898     0958  5                           TABLE[TBL$B_OLD_ID] = .REC_ADDR[IRC$B_RRV_ID];
;   899     0959  5
;   900     0960  5                       TABLE = .TABLE + 7;
;   901     0961  5                       END
```

```
;   902     0962  4                ELSE
;   903     0963  5                    BEGIN
;   904     0964  5                    TABLE[TBL$W_NEW_ID] = .NXTID;
;   905     0965  5
;   906     0966  5                    IF .FLAG[FLG$V_UPD_POS]
;   907     0967  5                    THEN
;   908     0968  5                        TABLE[TBL$W_OLD_ID] = .IRAB[IRB$W_PUTUP_ID]
;   909     0969  5                    ELSE
;   910     0970  6                        BEGIN
;   911     0971  6                        TABLE[TBL$W_OLD_ID] = .REC_ADDR[IRC$W_RRV_ID];
;   912     0972  6
;   913     0973  6                        ! If the current record was deleted within a Recovery
;   914     0974  6                        ! Unit, then save this information in the flag byte
;   915     0975  6                        ! of the table entry.
;   916     0976  6                        !
;   917     0977  6                        IF .REC_ADDR[IRC$V_RU_DELETE]
;   918     0978  6                        THEN
;   919     0979  6                            TABLE[TBL$V_RU_DELETE] = 1;
;   920     0980  5                        END;
;   921     0981  5
;   922     0982  5                    TABLE = .TABLE + 10;
;   923     0983  4                    END;
;   924     0984  4
;   925     0985  4                IF NOT .FLAG[FLG$V_UPD_POS]
;   926     0986  4                THEN
;   927     0987  4                    RMSGETNEXT_REC()
;   928     0988  4
;   929     0989  4                END             ! end of else record has moved before !
;   930     0990  4
;   931     0991  4            ! Else the current record is a deleted record, then just get the next
;   932     0992  4            ! record.  (Do not need to check FLG$V_UPD_POS, because on a bucket
;   933     0993  4            ! split because of no more id's available, it was on an insert oper-
;   934     0994  4            ! ation, not an update).
;   935     0995  4            !
;   936     0996  3            ELSE
;   937     0997  3                RMSGETNEXT_REC();
;   938     0998  3
;   939     0999  3            ! bump the nxtid
;   940     1000  3            !
;   941     1001  3            NXTID = .NXTID + 1;
;   942     1002  3
;   943     1003  3            ! clear the "at pos_for_insert in update mode" flag
;   944     1004  3            !
;   945     1005  3            FLAG[FLG$V_UPD_POS] = 0;
;   946     1006  3            END;                                    ! { end of while loop }
;   947     1007  2
;   948     1008  2        ! if there still are records that need to be squashed out, do it
;   949     1009  2        !
;   950     1010  2
;   951     1011  2        IF .SQUISH NEQU .REC_ADDR
;   952     1012  2        THEN
;   953     1013  3            BEGIN
;   954     1014  3            RMSSQUISH(.EOB, .SQUISH);
;   955     1015  3            EOB = .EOB - (.REC_ADDR - .SQUISH);
;   956     1016  3            REC_ADDR = .SQUISH;
;   957     1017  2            END;
;   958     1018  2
```

```
;      2        ! update the freespace word
;      2        !
;      2        BKT_ADDR[BKT$W_FREESPACE] = .EOB - .BKT_ADDR;
;      2
;      2        ! mark the end of the table in its first word for future reference
;      2        !
;      3        BEGIN
;      3
;      3        LOCAL
;      3            BEG_TABLE        : REF BBLOCK;
;      3
;      3        BEG_TABLE = .BBLOCK[.IRAB[IRB$L_NXTBDB], BDB$L_ADDR];
;      3        BEG_TABLE[TBL$W_FFB] = .TABLE - .BEG_TABLE
;      2        END;
;      2        RETURN;
;      2
;      1        END;                                    ! { end of routine }


                                 3C  BB 00000 RM$UPDATE_RRV::
                                                      PUSHR   #^M<R2,R3,R4,R5>                ; 0507
                      5E         1C  C2 00002         SUBL2   #28, SP
                                 7E  D4 00005         CLRL    FLAG                            ; 0580
                      50     3C  A9  D0 00007         MOVL    60(IRAB), R0                    ; 0581
       52        18   A0         02  C1 0000B         ADDL3   #2, 24(R0), TABLE
                      50     20  A9  D0 00010         MOVL    32(IRAB), R0                    ; 0582
                      55     18  A0  D0 00014         MOVL    24(R0), BKT_ADDR
                      56     4A  A9  3C 00018         MOVZWL  74(IRAB), REC_ADDR              ; 0583
                      56         55  C0 0001C         ADDL2   BKT_ADDR, REC_ADDR
                      53     04  A5  3C 0001F         MOVZWL  4(BKT_ADDR), EOB                ; 0584
                      53         55  C0 00023         ADDL2   BKT_ADDR, EOB
                      51     14  A0  3C 00026         MOVZWL  20(R0), R1                      ; 0585
                               6145  9F 0002A         PUSHAB  (R1)[BKT_ADDR]
                      03   00B7  CA  91 0002D         CMPB    183(IFAB), #3                   ; 0591
                                 02  12 00032         BNEQ    1$
                                 6E  D7 00034         DECL    REAL_END                        ; 0593
                      50     48  A9  3C 00036 1$:     MOVZWL  72(IRAB), R0                    ; 0595
  10   AE             55         50  C1 0003A         ADDL3   R0, BKT_ADDR, POS_INS
             08       AE         56  D0 0003F         MOVL    REC_ADDR, SQUISH                ; 0596
                      56     10  AE  D1 00043         CMPL    POS_INS, REC_ADDR               ; 0601
                                 04  1E 00047         BGEQU   2$
             04       AE         01  88 00049         BISB2   #1, FLAG                        ; 0603
                      56     10  AE  D1 0004D 2$:     CMPL    POS_INS, REC_ADDR               ; 0605
                                 09  12 00051         BNEQ    3$
       04    44       A9         03  E1 00053         BBC     #3, 68(IRAB), 3$                ; 0607
             04       AE         01  88 00058         BISB2   #1, FLAG                        ; 0609
       0E    44       A9         02  E1 0005C 3$:     BBC     #2, 68(IRAB), 4$                ; 0613
             14       AE   0090  C9  D0 00061         MOVL    144(IRAB), VBN                  ; 0616
             0C       AE   00A2  C9  B0 00067         MOVW    162(IRAB), NXTID                ; 0617
                                 0C  11 0006D         BRB     5$
             14       AE   008C  C9  D0 0006F 4$:     MOVL    140(IRAB), VBN                  ; 0621
             0C       AE   00A0  C9  B0 00075         MOVW    160(IRAB), NXTID                ; 0622
                      53         56  D1 0007B 5$:     CMPL    REC_ADDR, EOB                   ; 0632
                                 03  1B 0007E         BLEQU   7$                              ;
```

```
                                0254  31 00080 6$:      BRW     50$                                                     ;
                                  04  13 00083 7$:      BEQL    8$                                                      ; 0647
                09        66      03  E1 00085         BBC     #3, (REC_ADDR), 9$                                      ; 0649
                          F3  04    AE  E8 00089 8$:     BLBS    FLAG, 6$                                                ; 0652
                EE  06    A9        03  E1 0008D         BBC     #3, 6(IRAB), 6$                                         ; 0654
                                    0A  13 00092 9$:      BEQL    10$                                                     ; 0660
                06        66        02  E1 00094         BBC     #2, (REC_ADDR), 10$                                     ; 0662
                    04    AE        10  88 00098         BISB2   #16, FLAG                                               ; 0664
                                    04  11 0009C         BRB     11$                                                     ;
                    04    AE        10  8A 0009E 10$:     BICB2   #16, FLAG                                               ; 0666
                50  56              55  A3 000A2 11$:     SUBW3   BKT_ADDR, REC_ADDR, DIFFERENCE                          ; 0668
                53  44    A9        02  E1 000A6         BBC     #2, 68(IRAB), 15$                                       ; 0674
                    4C    A9        50  B1 000AB         CMPW    DIFFERENCE, 76(IRAB)                                    ; 0678
                                    32  12 000AF         BNEQ    14$                                                     ;
                2D  04    AE        01  E0 000B1         BBS     #1, FLAG, 14$                                           ; 0680
                          0A  04    AE  E8 000B6         BLBS    FLAG, 12$                                               ; 0683
                05  44    A9        03  E1 000BA         BBC     #3, 68(IRAB), 12$                                       ; 0685
                1F  06    A9        03  E0 000BF         BBS     #3, 6(IRAB), 14$                                        ; 0687
                    04    AE        02  88 000C4 12$:     BISB2   #2, FLAG                                                ; 0690
                    14    AE  70    A9  D0 000C8         MOVL    112(IRAB), VBN                                          ; 0694
                                    0E  12 000CD         BNEQ    13$                                                     ; 0699
                    14    AE  008C  C9  D0 000CF         MOVL    140(IRAB), VBN                                          ; 0702
                    0C    AE  00A0  C9  B0 000D5         MOVW    160(IRAB), NXTID                                        ; 0703
                                    06  11 000DB         BRB     14$                                                     ;
                    0C    AE  00A4  C9  B0 000DD 13$:     MOVW    164(IRAB), NXTID                                        ; 0706
                    4E    A9        50  B1 000E3 14$:     CMPW    DIFFERENCE, 78(IRAB)                                    ; 0710
                                    15  12 000E7         BNEQ    15$                                                     ;
                10  04    AE        02  E0 000E9         BBS     #2, FLAG, 15$                                           ; 0712
                    04    AE        04  88 000EE         BISB2   #4, FLAG                                                ; 0716
                    14    AE  008C  C9  D0 000F2         MOVL    140(IRAB), VBN                                          ; 0718
                    0C    AE  00A0  C9  B0 000F8         MOVW    160(IRAB), NXTID                                        ; 0719
                    18    AE    14  AE  D0 000FE 15$:     MOVL    VBN, VBN1                                               ; 0731
                    10    AE        56  D1 00103         CMPL    REC_ADDR, POS_INS                                       ; 0733
                                    38  12 00107         BNEQ    17$                                                     ;
                          34  04    AE  E8 00109         BLBS    FLAG, 17$                                               ; 0735
                    04    AE        01  88 0010D         BISB2   #1, FLAG                                                ; 0738
                1C  06    A9        03  E1 00111         BBC     #3, 6(IRAB), 16$                                        ; 0740
                    04    AE        08  88 00116         BISB2   #8, FLAG                                                ; 0743
                22  44    A9        02  E1 0011A         BBC     #2, 68(IRAB), 17$                                       ; 0745
                    4C    A9  4A    A9  B1 0011F         CMPW    74(IRAB), 76(IRAB)                                      ; 0747
                                    1B  12 00124         BNEQ    17$                                                     ;
                    04    AE        02  8A 00126         BICB2   #2, FLAG                                                ; 0750
                    18    AE  0090  C9  D0 0012A         MOVL    144(IRAB), VBN1                                         ; 0751
                                    0F  11 00130         BRB     17$                                                     ; 0745
                    4C    A9  4A    A9  B1 00132 16$:     CMPW    74(IRAB), 76(IRAB)                                      ; 0767
                                    08  13 00137         BEQL    17$                                                     ;
                03  44    A9        03  E0 00139         BBS     #3, 68(IRAB), 17$                                       ; 0769
                          0C    AE  B6  00013E          INCW    NXTID                                                   ; 0772
                          5C        03  D0 00141 17$:     MOVL    #3, AP                                                  ; 0776
                07  04    AE        03  E1 00144         BBC     #3, FLAG, 18$                                           ; 0783
                    20    AE  78    A9  D0 00149         MOVL    120(IRAB), RRV_VBN                                      ; 0785
                                    07  11 0014E         BRB     19$                                                     ;
                                0000G  30 00150 18$:     BSBW    RM$RECORD_VBN                                           ; 0787
                    20    AE        50  D0 00153         MOVL    R0, RRV_VBN                                             ;
                          50  20    A9  D0 00157 19$:     MOVL    32(IRAB), R0                                            ; 0796
                    1C    A0  20    AE  D1 0015B         CMPL    RRV_VBN, 28(R0)                                         ;
                                    03  13 00160         BEQL    21$                                                     ;
```

```
                                  0105  31 00162 20$:   BRW     41$                                           :
                   03     00B7      CA  91 00165 21$:   CMPB    183(IFAB), #3                                 : 0798
                                    0A  1F 0016A        BLSSU   22$                                           :
          05      04      AE        04  E1 0016C        BBC     #4, FLAG, 22$                                 : 0800
          EC      04      AE        03  E1 00171        BBC     #3, FLAG, 20$                                 : 0802
          08      04      AE        03  E1 00176 22$:   BBC     #3, FLAG, 23$                                 : 0809
                  1C      AE  0080  C9  B0 0017B        MOVW    128(IRAB), OLD_ID                             : 0811
                                    07  11 00181        BRB     24$                                           :
                                 0000G  30 00183 23$:   BSBW    RM$RECORD_ID                                  : 0813
                  1C      AE        50  B0 00186        MOVW    R0, OLD_ID                                    :
                          03  00B7  CA  91 0018A 24$:   CMPB    183(IFAB), #3                                 : 0815
                                    0F  1E 0018F        BGEQU   26$                                           :
          05      04      AE        04  E0 00191        BBS     #4, FLAG, 25$                                 : 0817
                          50        07  D0 00196        MOVL    #7, RRV_SIZE                                  : 0819
                                    08  11 00199        BRB     27$                                           :
                          50        02  D0 0019B 25$:   MOVL    #2, RRV_SIZE                                  : 0821
                                    03  11 0019E        BRB     27$                                           : 0817
                          50        09  D0 001A0 26$:   MOVL    #9, RRV_SIZE                                  : 0823
   51       04    AE      01        03  EF 001A3 27$:   EXTZV   #3, #1, FLAG, R1                              : 0832
                          51        51  D2 001A9        MCOML   R1, R1                                        :
                          50        53  C0 001AC        ADDL2   EOB, R0                                       : 0828
                          6E        50  D1 001AF        CMPL    R0, REAL_END                                  :
                          74        1F 001B2            BLSSU   35$                                           :
                   03               51  E9 001B4        BLBC    R1, 28$                                       : 0832
                                 0000G  30 001B7        BSBW    RM$GETNEXT_REC                                : 0834
                          54    08  AE  D0 001BA 28$:   MOVL    SQUISH, R4                                    : 0836
                                  FE30  30 001BE        BSBW    RM$SQUISH                                     :
          54      08      AE        56  C3 001C1        SUBL3   REC_ADDR, SQUISH, R4                          : 0837
                          53        54  C0 001C6        ADDL2   R4, EOB                                       :
          3C      44      A9        02  E1 001C9        BBC     #2, 68(IRAB), 32$                             : 0843
                          50    4C  A9  3C 001CE        MOVZWL  76(IRAB), R0                                  : 0847
                          50        55  C0 001D2        ADDL2   BKT_ADDR, R0                                  :
                          50    08  AE  D1 001D5        CMPL    SQUISH, R0                                    :
                                    11  1A 001D9        BGTRU   30$                                           :
                          56        50  D1 001DB        CMPL    R0, REC_ADDR                                  : 0851
                                    08  1A 001DE        BGTRU   29$                                           :
      4C  A9      08      AE        55  A3 001E0        SUBW3   BKT_ADDR, SQUISH, 76(IRAB)                    : 0853
                                    04  11 001E6        BRB     30$                                           :
                  4C      A9        54  A0 001E8 29$:   ADDW2   R4, 76(IRAB)                                  : 0856
                          50    4E  A9  3C 001EC 30$:   MOVZWL  78(IRAB), R0                                  : 0859
                          50        55  C0 001F0        ADDL2   BKT_ADDR, R0                                  :
                          50    08  AE  D1 001F3        CMPL    SQUISH, R0                                    :
                                    11  1A 001F7        BGTRU   32$                                           :
                          56        50  D1 001F9        CMPL    R0, REC_ADDR                                  : 0863
                                    08  1A 001FC        BGTRU   31$                                           :
      4E  A9      08      AE        55  A3 001FE        SUBW3   BKT_ADDR, SQUISH, 78(IRAB)                    : 0865
                                    04  11 00204        BRB     32$                                           :
                  4E      A9        54  A0 00206 31$:   ADDW2   R4, 78(IRAB)                                  : 0868
                  10      AE    08  AE  D1 0020A 32$:   CMPL    SQUISH, POS_INS                               : 0873
                                    11  1A 0020F        BGTRU   34$                                           :
                          56    10  AE  D1 00211        CMPL    POS_INS, REC_ADDR                             : 0877
                                    07  1A 00215        BGTRU   33$                                           :
                  10      AE    08  AE  D0 00217        MOVL    SQUISH, POS_INS                               : 0879
                                    04  11 0021C        BRB     34$                                           :
                  10      AE        54  C0 0021E 33$:   ADDL2   R4, POS_INS                                   : 0881
                          56    08  AE  D0 00222 34$:   MOVL    SQUISH, REC_ADDR                              : 0884
                                    06  11 00226        BRB     36$                                           : 0828
```

```
                                03              51  E9 00228 35$:    BLBC    R1, 36$                                 ; 0890
                                            0000G   30 0022B         BSBW    RM$GETNEXT_REC                          ; 0892
                                                63  94 0022E 36$:    CLRB    (EOB)                                   ; 0896
                                            63  08  88 00230         BISB2   #8, (EOB)                               ; 0897
                                            03  0087 CA 91 00233     CMPB    183(IFAB), #3                           ; 0899
                                                1D  1E 00238         BGEQU   38$
                    09      04  AE                  04  E1 0023A     BBC     #4, FLAG, 37$                           ; 0905
                                            83      14  88 0023F     BISB2   #20, (EOB)+                             ; 0909
                                            83  1C  AE  90 00242     MOVB    OLD_ID, (EOB)+                          ; 0910
                                                20  11 00246         BRB     40$                                     ; 0905
            83      02                      00      02  F0 00248 37$: INSV    #2, #0, #2, (EOB)+                      ; 0915
                                            83  1C  AE  90 0024D     MOVB    OLD_ID, (EOB)+                          ; 0916
                                            83  0C  AE  90 00251     MOVB    NXTID, (EOB)+                           ; 0917
                                                0D  11 00255         BRB     39$                                     ; 0918
            83      02                      00      02  F0 00257 38$: INSV    #2, #0, #2, (EOB)+                      ; 0924
                                            83  1C  AE  B0 0025C     MOVW    OLD_ID, (EOB)+                          ; 0925
                                            83  0C  AE  B0 00260     MOVW    NXTID, (EOB)+                           ; 0926
                                            83  18  AE  D0 00264 39$: MOVL    VBN1, (EOB)+                            ; 0927
                                                63  11 00268 40$:    BRB     49$                                     ; 0796
                    5B      04  AE                  04  E0 0026A 41$: BBS     #4, FLAG, 48$                           ; 0943
                                            18  AE  DD 0026F         PUSHL   VBN1                                    ; 0946
                                        FD31    30 00272             BSBW    RM$CODE_VBN
                                            5E      04  C0 00275     ADDL2   #4, SP
                                            62      50  90 00278     MOVB    R0, (TABLE)
                                01  A2          20  AE  D0 0027B     MOVL    RRV_VBN, 1(TABLE)                       ; 0947
                                            03  0087  CA 91 00280     CMPB    183(IFAB), #3                           ; 0949
                                                1C  1E 00285         BGEQU   44$
                                05  A2          0C  AE  90 00287     MOVB    NXTID, 5(TABLE)                         ; 0952
                    08          04  AE              03  E1 0028C     BBC     #3, FLAG, 42$                           ; 0956
                                06  A2      0080    C9  90 00291     MOVB    128(IRAB), 6(TABLE)
                                                05  11 00297         BRB     43$
                                06  A2          02  A6  90 00299 42$: MOVB    2(REC_ADDR), 6(TABLE)                   ; 0958
                                            52      07  C0 0029E 43$: ADDL2   #7, TABLE                              ; 0960
                                                22  11 002A1         BRB     47$                                     ; 0949
                                05  A2          0C  AE  B0 002A3 44$: MOVW    NXTID, 5(TABLE)                         ; 0964
                        08      04  AE              03  E1 002A8     BBC     #3, FLAG, 45$                           ; 0968
                                07  A2      0080    C9  B0 002AD     MOVW    128(IRAB), 7(TABLE)
                                                0D  11 002B3         BRB     46$
                                07  A2          03  A6  B0 002B5 45$: MOVW    3(REC_ADDR), 7(TABLE)                   ; 0971
                            04      66              05  E1 002BA     BBC     #5, (REC_ADDR), 46$                     ; 0977
                                09  A2              01  88 002BE     BISB2   #1, 9(TABLE)                            ; 0979
                                            52      0A  C0 002C2 46$: ADDL2   #10, TABLE                             ; 0982
                        03      04  AE              03  E0 002C5 47$: BBS     #3, FLAG, 49$                           ; 0985
                                                0000G   30 002CA 48$: BSBW    RM$GETNEXT_REC                          ; 0997
                                            0C  AE  B6 002CD 49$:    INCW    NXTID                                   ; 1001
                                04  AE              08  8A 002D0     BICB2   #8, FLAG                                ; 1005
                                                FDA4    31 002D4     BRW     5$                                      ; 0632
                                            56  08  AE  D1 002D7 50$: CMPL    SQUISH, REC_ADDR                        ; 1011
                                                13  13 002DB         BEQL    51$
                                            54  08  AE  D0 002DD     MOVL    SQUISH, R4                              ; 1014
                                                FDOD    30 002E1     BSBW    RM$SQUISH
                            54      08  AE              56  C3 002E4 SUBL3   REC_ADDR, SQUISH, R4                    ; 1015
                                            53      54  C0 002E9     ADDL2   R4, EOB
                                            56  08  AE  D0 002EC     MOVL    SQUISH, REC_ADDR                        ; 1016
                    04      A5              53      55  A3 002F0 51$: SUBW3   BKT_ADDR, EOB, 4(BKT_ADDR)              ; 1021
                                            50  3C  A9  D0 002F5     MOVL    60(IRAB), R0                            ; 1030
                                            50  18  A0  D0 002F9     MOVL    24(R0), BEG_TABLE                       ;
```

```
                 60        52        50  A3 002FD        SUBW3   BEG_TABLE, TABLE, (BEG_TABLE)                   ; 1031
                           5E        24  C0 00301        ADDL2   #36, SP                                         ; 1035
                                     3C  BA 00304        POPR    #^M<R2,R3,R4,R5>                                ;
                                         05 00306        RSB                                                     ;

; Routine Size:  775 bytes,    Routine Base:  RM$RMS3 + 005A

;  976          1036  1
```

```
;  978          1037  1 %SBTTL  'RM$UPDATE_RRV_2'
;  979          1038  1 GLOBAL ROUTINE RM$UPDATE_RRV_2 : RL$RABREG_4567 NOVALUE =
;  980          1039  1
;  981          1040  1 !++
;  982          1041  1 !
;  983          1042  1 ! FUNCTIONAL DESCRIPTION:
;  984          1043  1 !
;  985          1044  1 !     pdate the rrv's from other buckets.  Return with IRAB[IRB$V_RRV_ERR] set,
;  986          1045  1 !     f an error occurs during the update if it will cause the bucket to be trashed.
;  987          1046  1 !
;  988          1047  1 ! CALLING SEQUENCE:
;  989          1048  1 !       bsbw rm$update_2
;  990          1049  1 !
;  991          1050  1 ! INPUT PARAMETERS:
;  992          1051  1 !       NONE
;  993          1052  1 !
;  994          1053  1 ! IMPLICIT INPUTS:
;  995          1054  1 !       irab --
;  996          1055  1 !           nxtbdb -- referring to table of rrv's
;  997          1056  1 !           vbn_right, vbn_mid, rfa_vbn
;  998          1057  1 !           abovelckd - set when level 1 was locked coming down tree
;  999          1058  1 !       rab -- to store stv in
; 1000          1059  1 !       idx_dfn, IFAB, impure area, for rm$getbkt
; 1001          1060  1 !
; 1002          1061  1 ! OUTPUT PARAMETERS:
; 1003          1062  1 !       NONE
; 1004          1063  1 !
; 1005          1064  1 ! IMPLICIT OUTPUTS:
; 1006          1065  1 !       nxtbdb is released and cleared
; 1007          1066  1 !       rrv_err is set in the irab on any error
; 1008          1067  1 !
; 1009          1068  1 ! ROUTINE VALUE:
; 1010          1069  1 !       none -- rrv_err is set in the irab on any error
; 1011          1070  1 !               and the stv contains the actual status
; 1012          1071  1 !
; 1013          1072  1 ! SIDE EFFECTS:
; 1014          1073  1 !       rec_addr, ap, and bkt_addr are destroyed
; 1015          1074  1 !       nxtbdb is released and cleared
; 1016          1075  1 !       many buckets may be accessed and written out
; 1017          1076  1 !
; 1018          1077  1 !--
; 1019          1078  1
; 1020          1079  2     BEGIN
; 1021          1080  2
; 1022          1081  2     EXTERNAL REGISTER
; 1023          1082  2         COMMON_IO_STR,
; 1024          1083  2         R_REC_ADDR_STR,
; 1025          1084  2         COMMON_RAB_STR,
; 1026          1085  2         R_IDX_DFN_STR;
; 1027          1086  2
; 1028          1087  2     LOCAL
; 1029          1088  2         TABLE    : REF BBLOCK,
; 1030          1089  2         EOT;
; 1031          1090  2
; 1032          1091  2     LABEL
; 1033          1092  2         INNER,
; 1034          1093  2         INNERMOST,
```

```
;  1035          1094  2          BLK,
;  1036          1095  2          BLOCK;
;  1037          1096
;  1038          1097  BLOCK :
;  1039          1098      BEGIN
;  1040          1099
;  1041          1100      LOCAL
;  1042          1101          ENTRY_SIZE;
;  1043          1102
;  1044          1103      TABLE = .BBLOCK[.IRAB[IRB$L_NXTBDB], BDB$L_ADDR];
;  1045          1104      EOT = .TABLE + .TABLE[TBL$W_FFB];
;  1046          1105      TABLE = .TABLE + 2;
;  1047          1106
;  1048          1107      IF .IFAB[IFB$B_PLG_VER] LSSU PLG$C_VER_3
;  1049          1108      THEN
;  1050          1109          ENTRY_SIZE = 7
;  1051          1110      ELSE
;  1052          1111          ENTRY_SIZE = 10;
;  1053          1112
;  1054          1113      ! while there are still entries in the table, update each rrv individually
;  1055          1114      !
;  1056          1115
;  1057          1116      WHILE .TABLE LSSU .EOT
;  1058          1117      DO
;  1059          1118          BEGIN
;  1060          1119  4
;  1061          1120  4       ! if the table entry has already been taken care of, its vbn has
;  1062          1121  4       ! been cleared, so ignore it.
;  1063          1122  4       !
;  1064          1123  4
;  1065          1124  4       IF .TABLE[TBL$L_OLD_VBN] NEQ 0
;  1066          1125  4       THEN
;  1067          1126  4   INNER :
;  1068          1127  5           BEGIN
;  1069          1128  5
;  1070          1129  5           ! get the bucket to be updated
;  1071          1130  5           !
;  1072          1131  5       BLK :
;  1073          1132  6           BEGIN
;  1074          1133  6
;  1075          1134  6           LOCAL
;  1076          1135  6               ST,
;  1077          1136  6               SIZE;
;  1078          1137  6
;  1079          1138  6           SIZE = .IDX_DFN[IDX$B_DATBKTSZ]*512;
;  1080          1139  6           IRAB[IRB$B_CACHEFLGS] = CSH$M_LOCK;
;  1081          1140  6
;  1082          1141  6           ! if level above locked we must read the bucket with nowait to
;  1083          1142  6           ! avoid potential deadlock situation
;  1084          1143  6           !
;  1085          1144  6
;  1086          1145  6           IF .IRAB[IRB$V_ABOVELCKD]
;  1087          1146  6           THEN
;  1088          1147  6               BBLOCK[IRAB[IRB$B_CACHEFLGS], CSH$V_NOWAIT] = 1;
;  1089          1148  6
;  1090          1149  6           ST = RMSGETBKT(.TABLE[TBL$L_OLD_VBN], .SIZE);
;  1091          1150  6
```

```
;  1092          1151  6              IF .ST
;  1093          1152  6              THEN
;  1094          1153  6                  LEAVE BLK;
;  1095          1154  6
;  1096          1155  7              IF .ST<0, 16> EQL RMSERR(RLK)
;  1097          1156  6              THEN
;  1098          1157  7                  BEGIN
;  1099          1158  7
;  1100          1159  7                  ! we got a record lock error on the bucket so clear the flag
;  1101          1160  7                  ! and release the level 1 bucket to remove the deadlock
;  1102          1161  7                  ! potential
;  1103          1162  7                  !
;  1104          1163  7                  IRAB[IRB$V_ABOVELCKD] = 0;
;  1105          1164  7                  BDB = .IRAB[IRB$L_LOCK_BDB];
;  1106          1165  7                  IRAB[IRB$L_LOCK_BDB] = 0;
;  1107          1166  7                  RMSRLSBKT(0);
;  1108          1167  7
;  1109          1168  7                  ! re-read the bucket we want and wait for it this time
;  1110          1169  7                  !
;  1111          1170  7                  IRAB[IRB$B_CACHEFLGS] = CSH$M_LOCK;
;  1112          1171  7                  ST = RMSGETBKT(.TABLE[TBL$L_OLD_VBN], .SIZE);
;  1113          1172  7
;  1114          1173  7                  IF .ST
;  1115          1174  7                  THEN
;  1116          1175  7                      LEAVE BLK;
;  1117          1176  7
;  1118          1177  6                  END;
;  1119          1178  6
;  1120          1179  6              ! if here there was a hard failure on either the first or second
;  1121          1180  6              ! getbkt
;  1122          1181  6              !
;  1123          1182  6              RAB[RAB$L_STV] = .ST;
;  1124          1183  6              IRAB[IRB$V_RRV_ERR] = 1;
;  1125          1184  6              LEAVE INNER;
;  1126          1185  6
;  1127          1186  5              END;                                    ! of local ST
;  1128          1187  6              BEGIN
;  1129          1188  6
;  1130          1189  6              LOCAL
;  1131          1190  6                  PTR     : REF BBLOCK;
;  1132          1191  6
;  1133          1192  6              PTR = .TABLE;
;  1134          1193  6
;  1135          1194  6              ! Do all the rrv's in this bucket that we have accessed.  Scan
;  1136          1195  6              ! through the rest of the table, comparing vbn's if we find one that
;  1137          1196  6              ! is the same as this one, take care of it now
;  1138          1197  6              !
;  1139          1198  6
;  1140          1199  6              WHILE .PTR LSSU .EOT
;  1141          1200  6              DO
;  1142          1201  7                  BEGIN
;  1143          1202  7
;  1144          1203  7                  IF .PTR[TBL$L_OLD_VBN] EQLU .TABLE[TBL$L_OLD_VBN]
;  1145          1204  7                  THEN
;  1146          1205  7 INNERMOST :
;  1147          1206  8                      BEGIN
;  1148          1207  8
```

```
;  1149     1208  8                    BUILTIN
;  1150     1209  8                        AP;
;  1151     1210  8
;  1152     1211  8                    IF .IFAB[IFB$B_PLG_VER] LSSU PLG$C_VER_3
;  1153     1212  8                    THEN
;  1154     1213  8                        AP = .PTR[TBL$B_OLD_ID]
;  1155     1214  8                    ELSE
;  1156     1215  8                        AP = .PTR[TBL$W_OLD_ID];
;  1157     1216  8
;  1158     1217  9                    BEGIN
;  1159     1218  9
;  1160     1219  9                    LOCAL
;  1161     1220  9                        ST;
;  1162     1221  9
;  1163     1222  9                    ST = RMSFIND_BY_ID();
;  1164     1223  9
;  1165     1224  9                    ! If bad status returned (ex: could not find by RFA)
;  1166     1225  9                    ! or this is NOT an RRV, or it is a DELETED RRV,
;  1167     1226  9                    ! then indicate error and mark entry done.
;  1168     1227  9                    !
;  1169     1228  9                    IF NOT .ST
;  1170     1229  9                        OR
;  1171     1230  9                        NOT .REC_ADDR[IRC$V_RRV]
;  1172     1231 10                        OR (.REC_ADDR[IRC$V_RRV] AND .REC_ADDR[IRC$V_DELETED])
;  1173     1232  9                    THEN
;  1174     1233 10                        BEGIN
;  1175     1234 10
;  1176     1235 10                        ! Indicates that this table entry has been taken
;  1177     1236 10                        ! care of.
;  1178     1237 10                        !
;  1179     1238 10                        IF .PTR NEQ .TABLE
;  1180     1239 10                        THEN
;  1181     1240 10                            PTR[TBL$L_OLD_VBN] = 0;
;  1182     1241 10
;  1183     1242 10                        ! If the current table entry indicates that the
;  1184     1243 10                        ! corresponding record had not been deleted within a
;  1185     1244 10                        ! Recovery Unit, then as there must be a RRV for it
;  1186     1245 10                        ! somewhere, this inability to find one represents an
;  1187     1246 10                        ! error. Make sure that an RVU error will get returned
;  1188     1247 10                        ! in this case so the user knows to expect that some
;  1189     1248 10                        ! RRV pointers in the file will be incorrect.
;  1190     1249 10                        !
;  1191     1250 10                        IF NOT .PTR[TBL$V_RU_DELETE]
;  1192     1251 10                        THEN
;  1193     1252 11                            BEGIN
;  1194     1253 11                            RAB[RAB$L_STV] = .ST;
;  1195     1254 11                            IRAB[IRB$V_RRV_ERR] = 1;
;  1196     1255 10                            END;
;  1197     1256 10
;  1198     1257 10                        LEAVE INNERMOST;
;  1199     1258  9                        END;
;  1200     1259  9
;  1201     1260  8                    END;                        ! ( end of block defining st )
;  1202     1261  8
;  1203     1262  8                    IF .IFAB[IFB$B_PLG_VER] LSSU PLG$C_VER_3
;  1204     1263  8                    THEN
;  1205     1264  9                        BEGIN
```

```
;  1206     1265  9                    REC_ADDR[IRC$B_RRV_ID] = .PTR[TBL$B_NEW_ID];
;  1207     1266  9                    REC_ADDR[IRC$L_RRV_VBN] = RM$SELECT_VBNT.PTR[TBL$B_NEW_VBN]);
;  1208     1267  9                    END
;  1209     1268  8                ELSE
;  1210     1269  9                    BEGIN
;  1211     1270  9                    REC_ADDR[IRC$W_RRV_ID] = .PTR[TBL$W_NEW_ID];
;  1212     1271  9                    REC_ADDR[IR3$L_RRV_VBN] = RM$SELECT_VBNT.PTR[TBL$B_NEW_VBN]);
;  1213     1272  8                    END;
;  1214     1273  8
;  1215     1274  8                PTR[TBL$L_OLD_VBN] = 0;
;  1216     1275  7                END;                    ! { end of vbns match -- innermost }
;  1217     1276  7
;  1218     1277  7            PTR = .PTR + .ENTRY_SIZE;
;  1219     1278  6            END;                                ! {end of while loop }
;  1220     1279  6
;  1221     1280  5        END;                                    ! of local PTR
;  1222     1281  5
;  1223     1282  5        ! if we're done w/ this vbn, release it, writing it out
;  1224     1283  5        !
;  1225     1284  6        BEGIN
;  1226     1285  6        BDB[BDB$V_DRT] = 1;
;  1227     1286  7        BEGIN
;  1228     1287  7
;  1229     1288  7        LOCAL
;  1230     1289  7            ST;
;  1231     1290  7
;  1232     1291  8        IF NOT (ST = RM$RLSBKT(RLS$M_WR_THRU))
;  1233     1292  7        THEN
;  1234     1293  8            BEGIN
;  1235     1294  8            RAB[RAB$L_STV] = .ST;
;  1236     1295  8            IRAB[IRB$V_RRV_ERR] = 1;
;  1237     1296  8            LEAVE INNER
;  1238     1297  8
;  1239     1298  7            END;
;  1240     1299  7
;  1241     1300  6            END;        ! { end of block defining st for call to rlsbkt }
;  1242     1301  5            END;
;  1243     1302  4            END;                ! {end of table entry is valid -- inner }
;  1244     1303  4
;  1245     1304  4        TABLE = .TABLE + .ENTRY_SIZE;
;  1246     1305  3        END;                                    ! {end of while loop }
;  1247     1306  3
;  1248     1307  2    END;                                        ! { end of block }
;  1249     1308  2
;  1250     1309  2    ! Release the buffer we used as a work space can't use rm$rlsbkt since it
;  1251     1310  2    ! makes too many checks & i've clobbered the buffer
;  1252     1311  2    !
;  1253     1312  2    BDB = .IRAB[IRB$L_NXTBDB];
;  1254     1313  2    IRAB[IRB$L_NXTBDB] = 0;
;  1255     1314  2    BDB[BDB$B_FLGS] = 0;
;  1256     1315  2    RM$RELEASE(0);
;  1257     1316  1    END;
```

```
                                 0C  BB 00000 RM$UPDATE_RRV_2::
                                             PUSHR   #^M<R2,R3>                      ; 1038
                              5E     0C  C2 00002             SUBL2   #12, SP
                              50 3C  A9  D0 00005             MOVL    60(IRAB), R0    ; 1103
                              53 18  A0  D0 00009             MOVL    24(R0), TABLE
                              50     63  3C 0000D             MOVZWL  (TABLE), R0     ; 1104
                        04    AE     83  40 9E 00010          MOVAB   (TABLE)+[R0], EOT
                              53     D6  00015                INCL    TABLE           ; 1105
                              03 00B7 CA  91 00017            CMPB    183(IFAB), #3   ; 1107
                                 05  1E  0001C                BGEQU   1$
                              6E     07  D0 0001E             MOVL    #7, ENTRY_SIZE  ; 1109
                                 03  11  00021                BRB     2$
                              6E     0A  D0 00023 1$:         MOVL    #10, ENTRY_SIZE ; 1111
                        04    AE     53  D1 00026 2$:         CMPL    TABLE, EOT      ; 1116
                                 03  1F  0002A                BLSSU   3$
                              00FD   31  0002C                BRW     20$
                        08    AE  01 A3  D0 0002F 3$:         MOVL    1(TABLE), 8(SP) ; 1124
                                 03  12  00034                BNEQ    4$
                              00ED   31  00036                BRW     19$
                              52  17 A7  9A 00039 4$:         MOVZBL  23(IDX_DFN), SIZE ; 1138
                           52 52     09  78 0003D             ASHL    #9, SIZE, SIZE
                        40 A9        01  90 00041             MOVB    #1, 64(IRAB)    ; 1139
                     04 06 A9        05  E1 00045             BBC     #5, 6(IRAB), 5$ ; 1145
                        40 A9        02  88 0004A             BISB2   #2, 64(IRAB)    ; 1147
                              52     DD  0004E 5$:            PUSHL   SIZE            ; 1149
                           0C AE     DD  00050                PUSHL   12(SP)
                              0000G  30  00053                BSBW    RM$GETBKT
                              5E     08  C0 00056             ADDL2   #8, SP
                              51     50  D0 00059             MOVL    R0, ST
                              36     51  E8 0005C             BLBS    ST, 7$          ; 1151
                     82AA 8F          51  B1 0005F             CMPW    ST, #33450      ; 1155
                                 28  12  00064                BNEQ    6$
                        06 A9        20  8A 00066             BICB2   #32, 6(IRAB)    ; 1163
                              54  0084 C9  D0 0006A             MOVL    132(IRAB), BDB  ; 1164
                                  0084 C9  D4 0006F             CLRL    132(IRAB)       ; 1165
                                       7E  D4 00073             CLRL    -(SP)           ; 1166
                                 0000G 30  00075                BSBW    RM$RLSBKT
                        40 A9        01  90 00078             MOVB    #1, 64(IRAB)    ; 1170
                              6E     52  D0 0007C             MOVL    SIZE, (SP)      ; 1171
                                 0C  AE  DD 0007F             PUSHL   12(SP)
                                 0000G 30  00082              BSBW    RM$GETBKT
                              5E     08  C0 00085             ADDL2   #8, SP
                              51     50  D0 00088             MOVL    R0, ST
                              07     51  E8 0008B             BLBS    ST, 7$          ; 1173
                        0C    A8     51  D0 0008E 6$:         MOVL    ST, 12(RAB)     ; 1182
                                 008D 31  00092                BRW     18$             ; 1183
                              52     53  D0 00095 7$:         MOVL    TABLE, PTR      ; 1192
                        04    AE     52  D1 00098 8$:         CMPL    PTR, EOT        ; 1199
                                 71  1E  0009C                BGEQU   17$
                        08    AE  01 A2  D1 0009E             CMPL    1(PTR), 8(SP)   ; 1203
                                 65  12  000A3                BNEQ    16$
                              03 00B7 CA  91 000A5             CMPB    183(IFAB), #3   ; 1211
                                 06  1E  000AA                BGEQU   9$
                              5C  06 A2  9A 000AC             MOVZBL  6(PTR), AP      ; 1213
                                 04  11  000B0                BRB     10$
                              5C  07 A2  3C 000B2 9$:         MOVZWL  7(PTR), AP      ; 1215
                                 0000G 30  000B6 10$:         BSBW    RM$FIND_BY_ID   ; 1222
```

```
                08              50  E9 000B9          BLBC    ST, 11$                                 ; 1228
          04    66              03  E1 000BC          BBC     #3, (REC_ADDR), 11$                     ; 1230
          16    66              02  E1 000C0          BBC     #2, (REC_ADDR), 13$                     ; 1231
                53              52  D1 000C4 11$:     CMPL    PTR, TABLE                              ; 1238
                                03  13 000C7          BEQL    12$                                     ;
                            01  A2  D4 000C9          CLRL    1(PTR)                                  ; 1240
                3A          09  A2  E8 000CC 12$:     BLBS    9(PTR), 16$                             ; 1250
             0C A8                  50  D0 000D0          MOVL    ST, 12(RAB)                             ; 1253
             06 A9                  04  88 000D4          BISB2   #4, 6(IRAB)                             ; 1254
                                    30  11 000D8          BRB     16$                                     ; 1257
                03         00B7     CA  91 000DA 13$:     CMPB    183(IFAB), #3                           ; 1262
                                    14  1E 000DF          BGEQU   14$                                     ;
             02 A6          05  A2  90 000E1          MOVB    5(PTR), 2(REC_ADDR)                     ; 1265
                7E                  62  9A 000E6          MOVZBL  (PTR), -(SP)                            ; 1266
                                FBDB  30 000E9          BSBW    RM$SELECT_VBN                           ;
                5E                  04  C0 000EC          ADDL2   #4, SP                                  ;
             03 A6                  50  D0 000EF          MOVL    R0, 3(REC_ADDR)                         ;
                                    12  11 000F3          BRB     15$                                     ; 1262
             03 A6          05  A2  B0 000F5 14$:     MOVW    5(PTR), 3(REC_ADDR)                     ; 1270
                7E                  62  9A 000FA          MOVZBL  (PTR), -(SP)                            ; 1271
                                FBC7  30 000FD          BSBW    RM$SELECT_VBN                           ;
                5E                  04  C0 00100          ADDL2   #4, SP                                  ;
             05 A6                  50  D0 00103          MOVL    R0, 5(REC_ADDR)                         ;
                            01  A2  D4 00107 15$:     CLRL    1(PTR)                                  ; 1274
                52                  6E  C0 0010A 16$:     ADDL2   ENTRY_SIZE, PTR                         ; 1277
                                    89  11 0010D          BRB     8$                                      ; 1199
             0A A4                  02  88 0010F 17$:     BISB2   #2, 10(BDB)                             ; 1285
                                    02  DD 00113          PUSHL   #2                                      ; 1291
                                0000G   30 00115          BSBW    RM$RLSBKT                               ;
                5E                  04  C0 00118          ADDL2   #4, SP                                  ;
                08                  50  E8 0011B          BLBS    ST, 19$                                 ;
             0C A8                  50  D0 0011E          MOVL    ST, 12(RAB)                             ; 1294
             06 A9                  04  88 00122 18$:     BISB2   #4, 6(IRAB)                             ; 1295
                53                  6E  C0 00126 19$:     ADDL2   ENTRY_SIZE, TABLE                       ; 1304
                                FEFA  31 00129          BRW     2$                                      ; 1116
                54          3C      A9  D0 0012C 20$:     MOVL    60(IRAB), BDB                           ; 1312
                            3C      A9  D4 00130          CLRL    60(IRAB)                                ; 1313
                            0A      A4  94 00133          CLRB    10(BDB)                                 ; 1314
                                        53  D4 00136          CLRL    R3                                      ; 1315
                        00000000G       00  16 00138          JSB     RM$RELEASE                              ;
                5E                      0C  C0 0013E          ADDL2   #12, SP                                 ; 1316
                                        0C  BA 00141          POPR    #^M<R2,R3>                              ;
                                            05 00143          RSB                                             ;

; Routine Size:  324 bytes,    Routine Base:  RM$RMS3 + 0361

; 1258          1317  1
; 1259          1318  1 END
; 1260          1319  1
; 1261          1320  0 ELUDOM
```

```
;                                   PSECT SUMMARY
;
;       Name                    Bytes                   Attributes
;
;  RM$RMS3                      1189  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  GBL,  REL,  CON,  PIC,ALIGN(2)



;                               Library Statistics
;
;                                        -------- Symbols --------      Pages       Processing
;       File                             Total    Loaded   Percent      Mapped      Time
;
;  _$255$DUA28:[RMS.OBJ]RMS.L32;1         3109        78         2       154         00:00.4




;                                   COMMAND QUALIFIERS

;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:RM3RRV/OBJ=OBJ$:RM3RRV SRC$:RM3RRV/UPDATE=(ENH$:RM3RRV)

; Size:         1189 code + 0 data bytes
; Run Time:       00:29.7
; Elapsed Time:   00:57.4
; Lines/CPU Min:   2669
; Lexemes/CPU-Min: 17387
; Memory Used:  302 pages
; Compilation Complete
```

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

RM3PROBE
LIS

RM3PUTERR
LIS

RM3PUTUPD
LIS

RM3SITDXSP
LIS

RM3SPLUDR
LIS

RM3RRV
LIS

RM3ROOT
LIS

RM3PUT
LIS